SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

D I P L O M S K I R A D

Matija Kovačić

Zagreb, 2012.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

D I P L O M S K I R A D

Mentor:
Prof. dr. sc. Mario Essert

Student:
Matija Kovačić

Zagreb, 2012.

FAKULTET STROJARSTVA I BRODOGRADNJE

Povjerenstvo za diplomske i završne ispite

IZJAVA

Pod punom moralnom odgovornošću izjavljujem da sam rad radio samostalno koristeći se znanjem stečenim tijekom studija, te navedenom literaturom.

Najsrdačnije se zahvaljujem voditelju rada prof. dr. sc. Mariu Essertu na pružanju korisnih savjeta, te stručne pomoći pri izradi ovog rada.

Na kraju zahvaljujem i svojoj obitelji na ukazanom povjerenju, potpori i strpljenju.

Matija Kovačić

Zagreb, 2012.

SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

Središnje povjerenstvo za završne i diplomske ispite

Povjerenstvo za diplomske ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa:	
Ur.broj:	

DIPLOMSKI ZADATAK

Student: **Matija KOVAČIĆ** Mat. br.: 0035166035

Naslov rada na hrvatskom jeziku: **Mjerenje s Arduino kontrolerom u realnom vremenu preko Interneta**

Naslov rada na engleskom jeziku: **Real-time measurements using Arduino controller over Internet**

Opis zadatka:

Danas se uporabom specijaliziranih mikrokontrolerskih uređaja ostvaruju brojna industrijska mjerenja. Trend razvitka takvih ugradbenih mjernih stanica idem putem standardizacije hardware-a i software-a za upravljanje i dohvaćanja mjerne informacije putem Interneta. Među mikrokontrolerima opće namjene u širokom spektru primjena ističe se *Arduino* mikrokontroler (http://www.arduino.cc/), a među programskim jezicima za rad preko WEBa u različitim aplikacijama ističe se *Processing* (http://processingjs.org/).
Na elektroničkom sklopu za regulaciju temperature treba napraviti sustav za mjerenje i regulaciju temperature putem WEB-sučelja. To uključuje projektiranje i realiziraciju WEB server-a koji omogućuje vezu između korisnika i sklopa. Kao izlaznu jedinica za regulaciju i upravljanje koristiti „*Arduino Ethernet*" mikrokontroler koji komunicira s računalom preko serijske veze. Komunikaciju između korisnika i *Arduina* osmisliti pomoću WEB sučelja stvorenog programskim paketom *Processing*.

Za realizaciju diplomskog zadatka potrebno je:

1. Osmisliti WEB server temeljen na *Linux* operacijskom sustavu koji će omogućiti komunikaciju između korisnika i *Arduina*.
2. Isprogramirati *Arduino* (komunikacija sa senzorima i Internetom), te aplikaciju u *Processingu* da bi se ostvarila veza *Arduino-PC* i *PC-WEB*-server.
3. Na elektroničkom modulu *Arduino* osmisliti i realizirati *GPS* senzor (za globalnu poziciju i vrijeme) i senzor za mjerenje temperature. Načiniti GUI sučelje za upravljanje modula i udaljenog procesa preko WEBa.
4. Projektirati i izvesti WEB sučelje (*PHP/SQLite, JavaScript/jQuery/Ajax*) za pregled i obradu mjerenih podataka u realnom vremenu s jednog ili više uređaja koji mjere na različitim mjestima u različito vrijeme.
5. Koristiti vremensko-prostornu os (program *Timeline* iz MIT Simile projekta http://www.simile-widgets.org/timeline/) za prikaz mjernih podataka i to on-line, generiranih ili spremljenih u SQLite bazi.

Zadatak zadan:	Rok predaje rada:	Predviđeni datum obrane:
13. rujna 2012.	15. studenog 2012.	21. i 22. studenog 2012.

Zadatak zadao: Predsjednik Povjerenstva:

Prof.dr.sc. Mario Essert Prof. dr. sc. Franjo Cajner

Sadržaj

Popis slika

Popis tablica

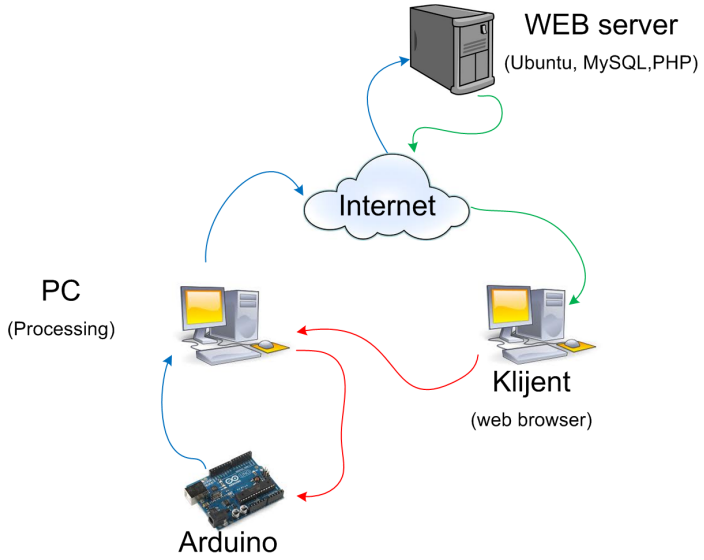
Sažetak

U ovom radu napravljeni je i osmišljeni sustav za upravljanje fizikalnim procesima putem web-a. Kao izlazna jedinica na računalu koristi se elektronički modul *Arduino*. Komunikacija između *Arduina* i računala osmišljena je putem *RS232* veze uz pomoć *Processing* aplikacije. Kako bi se ostvarila komunikacija između računala koje upravlja procesom i krajnjeg korisnika osmišljen je web server *Apache* na *Linux* operativnom sustavu. Upravljanje te kontrola procesa omogućena je putem sučelja u web browseru.

Ključne riječi: On-line aplikacije, processing, upravljanje, web upravljanje

Uvod

Cilj ovog diplomskog rada je osmisliti i sklopiti, uz pomoć *Arduina*, sustav koji će omogućiti mjerenje i upravljanje fizikalnim procesima na daljinu. Također je uz prikupljanje informacija potrebno osmisliti i pohranjivanje, prikazivanje te obradu informacija. Tako se omogućava krajnjem korisniku da provjerava stanje sustava kakvo je bilo prije te na temelju analize podataka donosi odluke kako dalje djelovati na sustav. Kako je danas prijenos informacija preko globalne internet mreže vrlo jednostavan i brz, sustav je osmišljen tako da se mjereni podaci šalju na jedno centralno mjesto (web server). Na taj način smo omogućili spremanje veće količine podataka i brži pristup informacijama zbog relacijskih baza podataka koje koristi web server.



Slika 1.1: Struktura komunikacije

Za samo mjerenje i djelovanje na fizikalni sustav koristit ćemo elektronički modul *Arduino*. *Arduino* u našem sustavu mjeri temperaturu okoline i temperaturu u postavu regulacije te GPS poziciju. Također preko svog PMW izlaza regulira temperaturu na postavu za regulaciju. Podatke o temperaturama i GPS poziciji *Arduino* šalje na osobno računalo s kojim je spojen sa USB (RS232) priključkom (sl. 1.1 – plava linija). Kako bi računalo primilo podatke napravljena je *Processing* aplikacija. Aplikacija omogućava primanje i obradu podataka primljenih

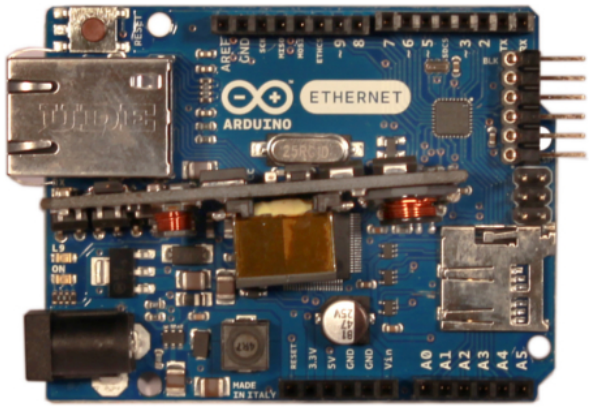
od *Arduina* te ispis istih na ekran. Nadalje kako bi podaci bili trajno spremljeni aplikacija iste podatke šalje na web server koji te podatke sprema u relacijsku bazu podataka.

Dio ovog zadatka je također samostalno odabrati, instalirati i podesiti web server. Za operacijski sustav web servera odabran je *Ubuntu Server Edition*. *Ubuntu* je vrlo jednostavan i rasprostranjen operacijski sustav te potpuno besplatan. Server sa osnovnom instalacijom podržava rad sa MySQL bazom podataka te PHP programskim jezikom. Pomoću PHP programskog jezika napravljeno je HTML sučelje koje omogućava krajnjem korisniku da pristupi mjerenim podacima, sa servera (sl. 1.1 – zelena linija), na vrlo brz, intuitivan i jednostavan način. Sučelje omogućava manipulaciju podacima, ljepši pregled mjerenih podataka te uspoređivanje podataka više mjernih stanica sa jednog mjesta.

Osim pregleda mjerenih podataka klijentu se omogućava djelovanje na *Arduino*. HTML sučelje klijentu nudi odabir temperature koju će postav za regulaciju održavati te također može uključiti ili isključiti regulaciju. U tom slučaju komunikacija teče u smjeru klijent-Arduino (sl. 1.1 – crvena linija). Klijent dobiva IP adresu računala na koje je spojen *Arduino* te tom računalu šalje željenu temperaturu ili zahtjev za isključenje/uključenje regulacije. Računalo nakon primitka nadalje šalje željenu temperaturu *Arduinu* putem USB priključka.

Elektronički modul “Arduino Ethernet”

Arduino ethernet je platforma otvorenog koda bazirana na mikrokontroleru *ATmega328* [1]. Ima 14 digitalnih ulazno/izlaznih pinova, 6 analognih pinova, radi na taktu 16 MHz. Prednost ove vrste korištenja mikrokontrolera je u tome što je on već opremljen sa svim komponentama potrebnim za rad. Jednostavno ga spojite na USB kabel i napajanje i spreman je za programiranje ili izvođenje nekog programa.



Slika 2.1: Elektronički modul “Arduino Ethernet”

Tablica 2.1: Tehničke karakteristike "Arduino Ethernet"

Mikrokontroler	ATmega328
Radni napon	5 V
Ulazni napon (preporučeno)	7-12 V
Ulazni napon (limit)	6-20 V
Digitalni I/O pinovi	14 (4 podržavaju PMW)
Analogni ulazni pinovi	6
DC struja po I/O pinu	40 mA
DC struja za 3.3V pin	50 mA
Flash memorija	32 KB (od čega 0.5 KB zauzima bootloader)
SRAM	2 KB
EEPROM	1 KB
Radni takt	16 MHz

Arduino se programira pomoću *Arduino* aplikacije (sl. 2.2) koja je također potpuno besplatna. Aplikacija nudi mogućnost programiranja *Arduina* preko serijske veze ako je na arduinu instaliran *boot loader*. *Boot loader* zauzima određeni dio programske memorije *Arduina*. Ukoliko vaš program zbog prevelike veličine ne bi stao u memoriju mikrokontrolera moguće je obrisati *boot loader*. U tom slučaju aplikacija nudi mogućnost programiranja preko *AVR ISP*, *Arduino as ISP*, *USBtinyISP*, *AVRISP mkII* ili *USBasp* modula.

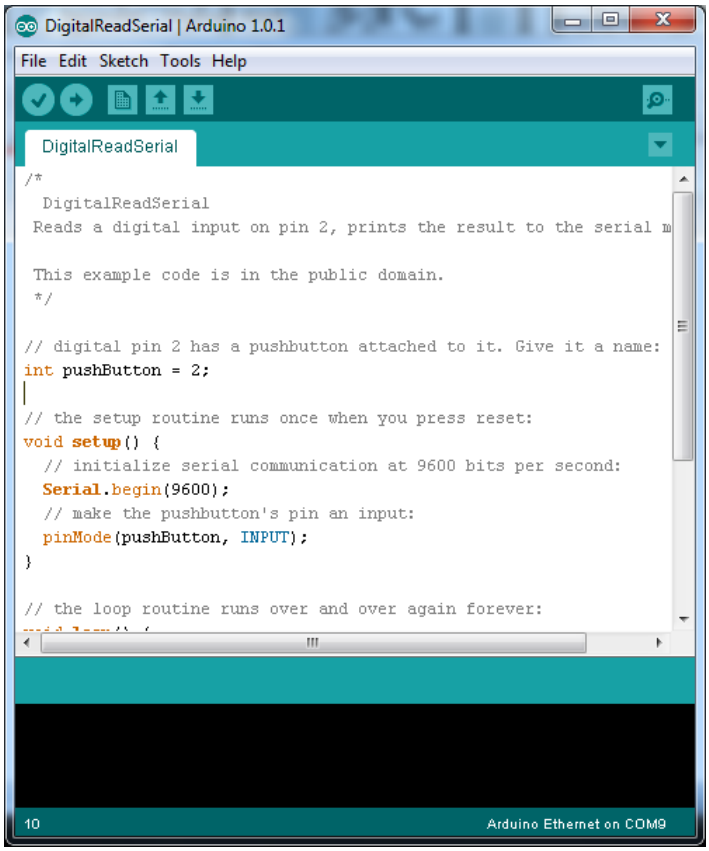
2.1 Funkcije setup() i loop()

Osnovna struktura programa za *Arduino* sastoji se od dvije glavne funkcije `setup()` i `loop()`. Funkcija `setup()` poziva se uvijek na početku pokretanja programa. U njoj se inicijaliziraju varijable, pokreću korišteni dodaci i slično. Funkcija `setup()` pokreče se samo jedanput nakon uključivanja ili resetiranja *Arduina*. Funkcija `loop()` sadržava programski kod koji se uvijek ponovo ponavlja tako omogućavajući aktivno kontroliranje *Arduina*. Svako ponavljanje `loop()` funkcije može imati drukčiji ishod koji može ovisiti o više različitih uvjeta npr. stanje ulaza na *Arduinu*, vremenska ovisnost, stanje primljenih podataka putem mreže itd.

Slijedi jednostavan primjer na kojem se može vidjeti struktura programa i uključivanje pina na *Arduinu*. Program gasi i uključuje pin u razmacima od jedne sekunde.

```
int led = 13;

//ovo je komentar :)
// funkcija setup pokrece se samo jedanput
void setup() {
```



Slika 2.2: Sučelje Arduina

```
  // inicijaliziramo pin-a br. 13 kao izlazni
  pinMode(led, OUTPUT);
}

// funkcija loop ponavlja se uvijek ponovno
void loop() {
  digitalWrite(led, HIGH);    // ukljucuje pin visoko (5V)
  delay(1000);                // cekaj sekundu
  digitalWrite(led, LOW);     // iskljucuje pin (0V)
  delay(1000);                // cekaj sekundu
}
```

2.2 Varijable

Kod mikrokontrolera *ATmega328* kao i kod svakog mikrokontrolera vrlo je važno točno deklarirati varijable. U skladu s deklaracijom varijable mikrokontroler zna koliko memorije mora rezervirati za tu varijablu. Tako sa dobrom deklaracijom varijabli možemo uštedjeti na memoriji ali zato istodobno moramo paziti na problem preljeva. Problem preljeva se javlja kad

primjerice u varijablu deklariranu kao `int` želimo upisati broj veći od 32767.

Tablica 2.2: Cjelobrojni tipovi podataka

TIP	**Veličina (bit)**	**Veličina (byte)**	**Minimalna vrijednost**	**Maksimalna vrijednost**
unsignet byte	8	1	0	255
byte	8	1	-128	127
unsignet int	16	2	0	65535
int	16	2	-32768	32767
unsignet long	32	4	0	4294967295
long	32	4	-2147483648	2147483647

Osim gore navedenih cjelobrojnih tipova podataka *Arduino* podržava i standardne tipove podataka poput *boolean*, *char*, *float*, *double*, *string*, *array* itd. *Boolean* tip podataka zauzima samo 1 bit memorije i može predstavljati samo dva stanja (istina, laž). *Char* zauzima jedan bajt memorije i predstavlja zapis *ASCII* znakova u obliku cjelobrojnog broja. *Float* predstavlja broj sa decimalnom točkom. Takav zapis zauzima (4 bajta) memorije. *Array* je kolekcija varijabli kojima se može pristupiti uz pomoć indeksa. *String* je kod *Arduina* zapravo *array* sastavljen od više *char* varijabli.

Kako bi koristili varijablu potrebno ju je deklarirati, postoje dva načina:

```
int inputVariable1;
int inputVariable2 = 0;
```

Prvi način prikazuje deklaraciju, a drugi deklaraciju i pridruživanje. Varijablu je moguće deklarirati unutar funkcije `setup()` te će tada ona biti globalna varijabla. Ako ne želimo da varijabla bude globalna tada varijablu možemo deklarirati unutar neke funkcije ali tada će biti dostupna samo unutar te funkcije.

Varijabla se koristi na način da joj se pridruži određena vrijednost koju želimo spremiti u varijablu. Pridruživanje vrijednosti varijabli vrši se pomoću znaka jednakosti:

```
inputVariable1 = 7;
inputVariable2 = analogRead(2);
```

Varijabli se može pridružiti neki broj npr. 7 kao iz gornjeg primjera ili se može pridružiti digitalizirana vrijednost napona na pinu (učitavanje analognog senzora). Također se uz pomoć `digitalRead()` može učitati vrijednost digitalnog pina.

2.3 Kontrola toka programa

Najčešće korištena naredba za kontrolu toka programa je `if`. Sintaksa naredbe je vrlo jednostavna:

```
If (varijabla>50)
{
        digitalWrite(LEDpin, HIGH)
}
else if (varijabl<50)
{
        digitalWrite(LEDpin, LOW)
}
else
{
        //naredbe C;
}
```

Napisani kod predstavlja program koji testira dali je brojčana vrijednost spremljena u "varijabli" veća od 50. Ako je varijabla veća od 50 program pin imenovan "LEDpin" diže na visoki nivo napona inače provjerava sljedeći uvjet. Ako ni sljedeći uvjet ne zadovoljava ući ćemo u "naredbe c;". Dakle uvjet za ulazak u `if` petlju je logičkog tipa (*boolean*). Ako je uvjet istinit ulazimo u petlju inače ju preskačemo. Naredba `if` može se koristiti u skraćeno verziji bez `else if` ili `else` ili oboje.

Kod naredbi za kontrolu toka često se koriste operatori usporedbe. U gornjem programu smo koristili operator usporedbe (>) veći od, također postoje i (<) manji od, (<=) manje ili jednako, (>=) veće ili jednako, (==) jednako i (!=) različito.

Slijedeća važna naredba u ovom poglavlju je `for` ona se koristi za ponavljanje bloka naredbi. Sintaksa naredbe je sljedeća:

```
for (inicijalizacija; uvjet; pribrojavanje) {
        //naredbe;
}
```

U argumentima naredbe `for` imamo tri argumenta. Prvi argument je "inicijalizacija" varijable o kojoj ovisi ulazak u petlju, "uvjet" ulaska u petlju, "pribrojavanje" promjena varijable o kojoj ovisi ulazak u petlju.

```
for (int i=0; i <= 255; i++){
   analogWrite(PWMpin, i);
   delay(10);
```

```
}
```

Ovaj primjer vrti blok naredbi 256 puta jer prvi put ulazi u petlju sa `i=0`, te svaki put varijabli `i` pribroji jedan. Program u svakom ponavljanju mijenja analognu vrijednost na svom pinu "PWMpin" i daje mu vrijednost veličine varijable `i`. Nakon pridodane vrijednosti program čeka 10 ms. Nakon 256 ponavljanja vrijednost varijable `i` prelazit će 255 i program više neće ponavljati blok naredbi.

2.4 Rad sa serijskom vezom

Serijsku vezu osim za programiranje Arduina koristimo i za komunikaciju između Arduina i računala ili nekog drugog uređaja. Arduino ima serijski port znan kao UART. On komunicira pomoću dva pina, pin 0 (RX) i pin 1 (TX). Dakle ako koristite serijsku vezu u radu svog *Arduina* tada pinove 0 i 1 ne možete koristiti u druge svrhe. Pokretanje serijske veze vrši se naredbom:

```
Serial.begin(9600);
```

Argument naredbe je brzina prijenosa serijskom vezom. Nakon pokretanja serijske veze uz pomoć naredbe `available()` možemo provjeriti koliko smo znakova serijskom vezom primili. Ako je broj znakova koje smo primili veći od 0 tada znakove možemo preuzeti s naredbom `read()`. Slanje podataka je vrlo jednostavno i vrši se naredbom `print()`. Ako nam veza više nije potrebna možemo je zaustaviti sa naredbom `end()`. Ovo je primjer programa koji prima podatak serijom i odmah ga šalje natrag uz rečenicu "Primio sam:":

```
int incomingByte = 0;

void setup() {
        Serial.begin(9600);
}

void loop() {
        if (Serial.available() > 0) {
                incomingByte = Serial.read();
                Serial.print("Primio sam: ");
                Serial.print(incomingByte, DEC);
        }
}
```

2.4.1 Programska serijska komunikacija

Arduino ima ugrađenu podršku za serijsku komunikaciju na pinovima 0 i 1, ali ponekad nam zatreba više serijskih portova. Kako bi omogućili serijski port na ostalim pinovima može na poslužiti `SoftwareSerial` biblioteka, ona koristi programsku implementaciju serijske veze. Ako koristite više softverskih portova samo jedan može komunicirati u određeno vrijeme, a ostali moraju čekati. Pokretanje programske serijske veze:

```
SoftwareSerial mySerial(10, 11); // RX, TX
```

Prvi argument je broj pina koji koristimo kao RX liniju, a drugi argument broj pina koji koristimo kao TX liniju. Naš novi port sad se naziva `mySerial` i pokrećemo ga sa naredbom `mySerial.begin(4800);`. Programski serijski port podržava iste naredbe poput hardverskog porta.

2.5 Samostalni pristup internetu

Arduino Ethernet posjeduje priključak za LAN kabel. Uz pomoć tog priključka *Arduino* može samostalno pristupiti LAN mreži ili priključenjem LAN kabla u ruter može pristupiti globalnoj internet mreži. *Arduino* pinovi koji su korišteni za ethernet su 10,11,12 i 13 stoga njih ne možemo koristiti u druge svrhe kad upotrebljavamo ethernet komunikaciju tokom rada *Arduina*. Za rad za ethernetom u *Arduino* programu potrebno je uključiti dodatak za ethernet `#include <Ethernet.h>`. Rad sa ethernetom započinje sa pokretanjem `Ethernet.begin(mac, ip, dns, gateway, subnet);` gdje je moguće izostaviti sve argumente osim `mac` argumenta. Opis argumenata:

- **mac** MAC (Media access control) je adresa ethernet adaptera vašeg *Arduino* uređaja, navedena je na naljepnici *Arduina*,
- **ip** je IP adresa vašeg uređaja. Ne morate ju navesti ako koristite gateway uređaj sa DHCP (Dynamic Host Configuration Protocol),
- **gateway** IP adresa vašeg gateway uređaja,
- **subnet** maska mreže.

Dodatak ima dvije klase `Client` i `Server`. Klasa `Client` se koristi za kreiranje klijenta na *Arduinu* koji se spaja na neki vanjski server i razmjenjuje podatke sa njim. Klasa `Server` se koristi za kreiranje servera na *Arduinu* koji šalje i prima podatke od vanjskih klijenata koji se spajaju na *Arduino*.

Slijedi primjer korištenja klase `Client`, spajanje arduina na google tražilicu sa postavljenim upitom "arduino". Primljeni odgovor *Arduino* serijskom vezom prosljeđuje na računalo. Odgovor upita je HTML kod web stranice google.

```
#include <SPI.h>
#include <Ethernet.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
char server[] = "www.google.hr";
EthernetClient client;

void setup()
{
  Ethernet.begin(mac);
  Serial.begin(9600);
  delay(1000);
  Serial.println("connecting...");
  if (client.connect(server,80)) {
    Serial.println("connected");
    client.println("GET /search?q=arduino HTTP/1.0");
    client.println();
  } else {
    Serial.println("connection failed");
  }
}

void loop()
{
  if (client.available()) {
    char c = client.read();
    Serial.print(c);
  }
  if (!client.connected()) {
    Serial.println();
    Serial.println("disconnecting.");
    client.stop();
    for(;;)
      ;
  }
}
```

Nakon uključivanja potrebnih dodataka slijedi definiranje MAC adrese i adrese servera na koji se želimo spojiti, slijedeće definiramo klasu klijent. Inicijalne postavke pokrećemo u `setup()` funkciji, a to su pokretanje etherneta te pokretanje serijske komunikacije. Zatim kroz `if` petlju spajamo našeg klijenta sa portom 80 na definirani server. Prilikom spajanja na server šaljemo GET upit `/search?q=arduino` taj nam upit vraća pretragu za riječ "arduino". U `loop()` funkciji provjeravamo ako je klijent dobio odgovor ako je dobio ispisujemo odgovor na serijski port. Slijedi još provjera konekcije klijenta ako veza više nije uspostavljena zaustavljamo našeg klijenta.

Sljedeći je primjer korištenje klase `Server`. Primjer prikazuje čekanje upita od klijenta, a kad primi upit klijentu natrag šalje isti tekst koji prima.

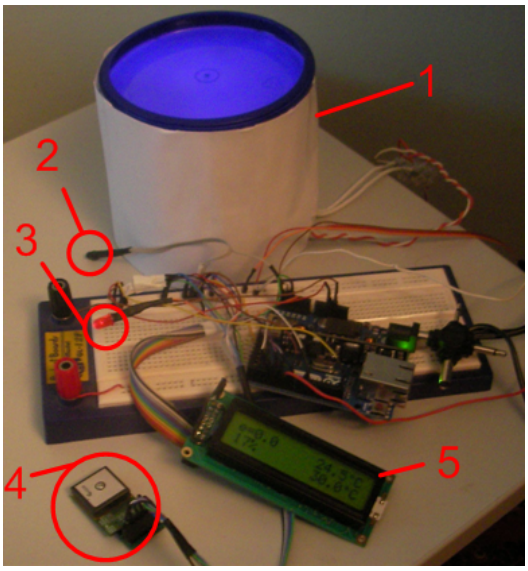
```
#include <SPI.h>
#include <Ethernet.h>

byte mac[] = {0x90, 0xA2, 0xDA, 0x00, 0x90, 0x41};
EthernetServer server(33);

void setup()
{
  Ethernet.begin(mac);
  server.begin();
}

void loop()
{
  EthernetClient client = server.available();
  if (client) {
    server.write(client.read());
  }
}
```

Nakon uključivanja modula slijedi definiranje MAC adrese i definiranje porta, kod definiranja porta važno je upisati argument koji predstavlja broj porta koji će server slušati. U `setup()` funkciji pokrećemo ethernet i server. Nadalje neprestano provjeravamo ako imamo dostupnog klijenta kad ga imamo tada od njega čitamo `client.read()` poruku i istu šaljemo natrag `server.write()`.



Slika 2.3: Moduli spojeni na “Arduino Ethernet”

2.6 Realizacija sklopa

Slika (sl. 2.3) predstavlja sklop za mjerenje upotrebljavan u ovom diplomskom radu. Kako bi realizirali sustav na *Arduino* pločicu potrebno je spojiti nekoliko modula:

- **(1)** postav za regulaciju temperature. Postav se sastoji od termometra i grijača. U postavu se može mijenjati, na daljinu putem web sučelja, referentna temperatura regulacije.
- **(2)** termometar, identični termometar posjeduje i postav za regulaciju. Ovaj vanjski služi za mjerenje temperature izvana.
- **(3)** LED dioda, služi kao indikator rada grijača ako je snaga grijača 0% dioda ne svijetli inače svijetli.
- **(4)** GPS senzor služi za automatsku lokaciju mjerne stanice.
- **(5)** LCD ekran služi za ispis osnovnih informacija o radu mjerne stanice.

U nastavku je opisan svaki pojedini modul sa shemom spajanja na *Arduino*. Opisan je i način na koji se vrši interakcija *Arduina* i modula kao i način programiranja kroz kod koji se upotrebljava u našem sustavu.

2.6.1 GPS-SKM53

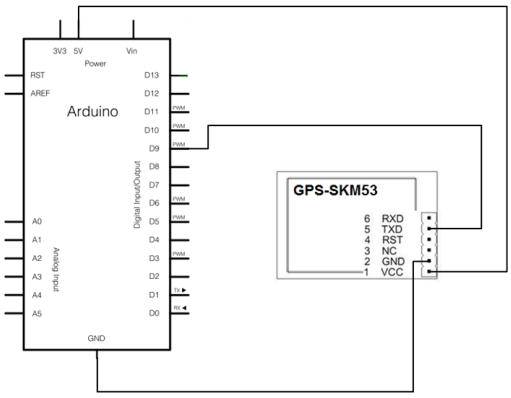
Za lociranje mjerne stanice u radu poslužio je čip SKM53 sa integriranom GPS antenom. SKM53 posjeduje 6 pinova od kojih su u radu korištena tri pina. Korišteni pinovi su TXD, VCC i GND. Putem TXD signala serijskom vezom modul šalje informacije pozicije, brzine te

vremena. Kako *Arduino* upotrebljava serijsku vezu za komunikaciju s računalom potrebno je otvoriti novi programski serijski port pomoću kojeg *Arduino* komunicira s GPS modulom.

Slika 2.4: GPS modul "SKM53"

TXD linija GPS-a spojena je na pin 9 *Arduina* te na taj pin treba inicirati programski serijski port `SoftwareSerial mySerial(9,8);`. Zadana brzina komunikacije GPS modula je 9600 bit/s pa uspostavljamo serijsku vezu `mySerial.begin(9600);`. Sve što primimo od GPS modula neposredno šaljemo na računalo:

```
if (mySerial.available()){
        Serial.write(mySerial.read());
}
```

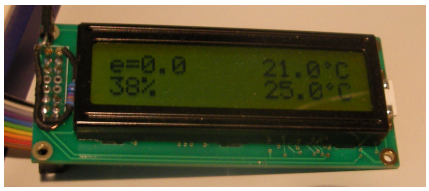


Slika 2.5: Shema spajanja GPS modula

2.6.2 LCD-GDM1602E

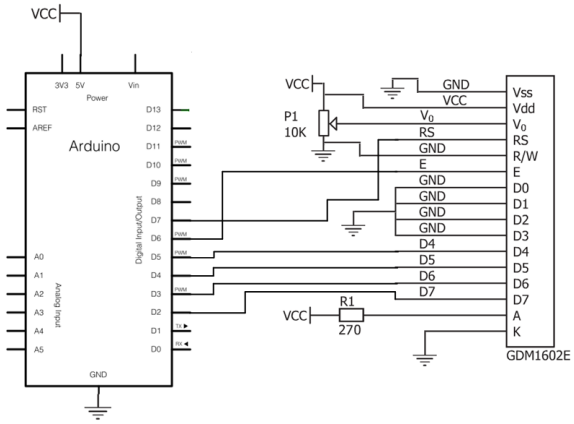
LCD ekran GDM1602E ima dva reda za prikaz znakova, u svakom redu može prikazati 16 znakova. LCD sastoji se od mikrokontrolera koji obrađuje primljene naredbe od drugog mikrkontrolera ili računala, LCD pogonskog sklopa čija je uloga upravljanje ispisom znakova na sam LCD ekran, LCD ekran i LED BKL koji služi za osvjetljenje ekrana. LCD modul ima 16 pinova od kojih se 8 pinova koristi za paralelni prijenos podataka na LCD modul.

Za prijenos podataka moguće je koristiti i 4 pinski prijenos podataka te je on korišten zbog uštede na broju pinova. U tom slučaju 4 preostala (DATA) pina spajamo na GND. Na pinovima



Slika 2.6: LCD modul “GDM1602E”

15 (A) i 16 (K) priključena je LED dioda koja daje pozadinsko osvjetljenje LCD-a, na njih spajamo otpornik u vrijednosti ovisnoj o tome kakvo osvjetljenje želimo. Signal *Vo* mijenja kontrast LCD ekrana te smo na pin *Vo* stavili potenciometar i tako smo omogućili da kontrast možemo mijenjati preko potenciometra. Signal *E* spajamo na arduinov pin br. 6, a signal *RS* na pin br. 7 i signal *R/W* na GND.



Slika 2.7: Shema spajanja LCD modula

Za rad sa LCD ekranom korištena je biblioteka `LiquidCrystal`. Biblioteka zahtijeva definiranje objekta `LiquidCrystal` prema tome kako je spojen LCD sklop:

```
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);
```

Sada možemo koristiti LCD ekran i ispisati temperaturu:

```
lcd.clear();              //brise ekran
lcd.setCursor(10,0);      //postavlja kursor u prvi red i 11 stupac
lcd.print(temp1,1);       //ispisuje varijablu u kojoj je trenutna temp.
lcd.print((char)178);     //ispisuje znak stupnja
lcd.print("C");           //ispisuje znak C

lcd.setCursor(10,1);      //postavlja kursor u drugi red i 11 stupac
lcd.print(temp2,1);       //ispisuje varijablu u kojoj je reg. temp.
lcd.print((char)178);
lcd.print("C");
```

```
lcd.setCursor(0,0);     //postavlja kursor u prvi red i 1 stupac
lcd.print("e=");
lcd.print(greska,1);    //ispisuje varijablu u kojoj je reg. greska

lcd.setCursor(0,1);     //postavlja kursor u drugi red i 1 stupac
lcd.print(int(snaga_delta));  //ispisuje varijablu u kojoj je snaga grijaca
lcd.print("%");
```

Na LCD ekran ispisujemo četiri osnovne informacije iz našeg sustava:

- **lijevo-gore** greška regulacije,
- **desno-gore** temperatura okoline,
- **lijevo-dole** snaga grijača,
- **desno-dole** temperatura u postavu.

2.6.3 Senzor temperature "LM35"

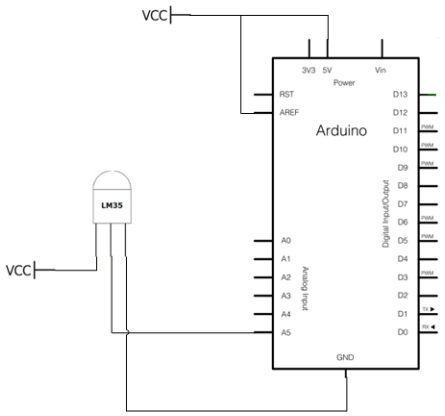
LM35 je analogni temperaturni senzor izvedeni kao integrirani krug. Senzor ima linearnu ovisnost izlaznog napona o temperaturi izraženoj u stupnjevima celzijusa. Točnost senzora bez kalibracije je unutar ± 1/4 °C na sobnoj temperaturi i ± 3/4 °C u punom rasponu -55 do +150 °C. Senzor vuče vrlo malo struje, samo 60 μA iz njegovog izvora, također se vrlo malo zagrijava manje od 0.1°C na mirnom zraku. Senzori su dostupni u hermetičkim *TO-46* kućištima, također i u *TO-92* tranzistorskim kućištima.

Slika 2.8: Senzor temperature "LM35", *TO-92* kućište

U našem slučaju koristimo dva temperaturna senzora jedan za mjerenje temperature okoline i jedan za mjerenje temperature u postavu za regulaciju temperature. Senzore spajamo na neke od analognih pinova u našem slučaju na pinove A5 (temp. okoline) i A4. Vrijednost napona na pinu učitavamo sa naredbom `analogRead(A5);`. Naredba nam vraća razinu napona na pinu u cjelobrojnom rasponu od 0-1024 (10 bit). Gdje je 1024 razina referentnog analognog napona. U našem slučaju stavili smo referentni napon 5 V što znači da jedan cjelobrojni dio

skale vrijedi 0.0049 V (4.9 mV). Linearno pojačanje LM35 senzora iznosi 0.01 $V/°C$ iz čega slijedi:

```
int sensorValue = analogRead(A5);            // npr. 45
float vrijednost_podjele=5.0/1024;           //0.0049
float temp_volt= vrijednost_podjele*sensorValue;      //0.0049*45=0.2205
float temp = temp_volt*1/0.01;               //0.2205*1/0.01 =22.05 C
```



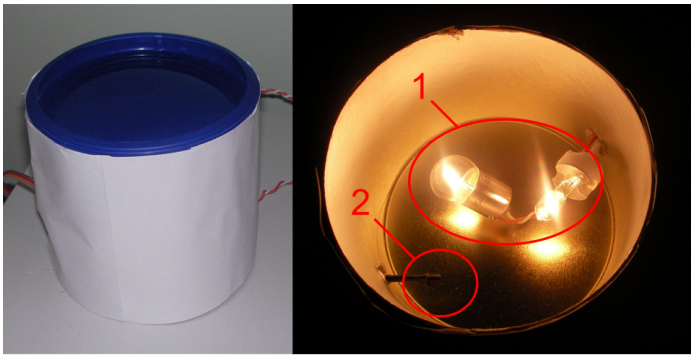
Slika 2.9: Shema spajanja senzora “LM35”

2.6.4 Postav za regulaciju temperature

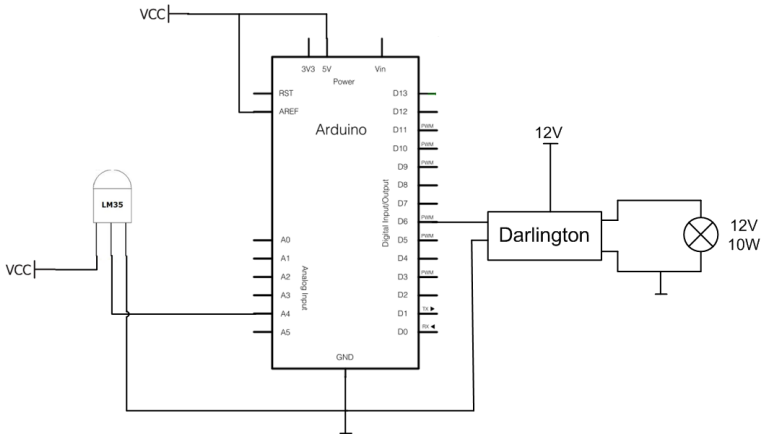
Postav za regulaciju temperature u našem sustavu sastoji se od valjkaste zatvorene posude volumena 785 cm^3. Kako bi zatvorili regulacijsku petlju u postavu za senzor temperature koristimo senzor opisan u prethodnom poglavlju. Kao grijač u našem postavu poslužit će dvije žarulje (12 V 5 W) koje pokreće *darlingtonovo* pojačalo. *Darlingtonovo* pojačalo spojeno je na *Arduinov* pin broj 6 koji podržava PWM [1] način rada. Napajanje za *darlingtonvo* pojačalo priskrbljuje odvojeni adapter izlazne struje 1 A i napona 12 V što je dovoljno za naš grijač.

PWM je tehnika kojom možemo dobiti prividni analogni napon na digitalnom izlazu. Digitalni izlaz ima dva moguća stanja visoko (5 V) i nisko (0 V) ako ta dva stanja izmjenjujemo visokom frekvencijom sa jednakom duljinom trajanja dobiti ćemo na digitalnom izlazu vrijednost napona od 2.5 V. Željeni izlazni napon možemo podešavati u rasponu od 0 V do 5 V na način da podešavamo omjer vremenskog trajanja visokog odnosno niskog stanja. PWM se sa *Arduinom* poziva naredbom `analogWrite(pin, vrijednost)` gdje je prvi argument pin na koji postavljamo analognu vrijednost, a drugi argument analogna vrijednost koju želimo postaviti. Analogna vrijednost se postavlja u rasponu od 0 do 255 gdje 255 predstavlja 5 V.

[1] Pulse Width Modulation – pulsno širinska modulacija



Slika 2.10: Postav za regulaciju a) postav izvana, b) postav iznutra (1-grijač 2-termometar)



Slika 2.11: Postav za regulaciju shema spajanja

```
if( (millis() - lct_r) > 100) {
    int sensorValue1 = analogRead(A4);
    float temp1=5.0/1024*sensorValue1*100;
    temp1=zaokruzi_temp(temp1);
    greska=reg_temp-temp1;
    float p_poj=1.5;          //pojacanje P
    float d_poj=2;            //pjacanje D
    float i_poj=0.07;         //pojacanje I
    float derivacija=(greska-greska_p)/100;
    float integracija=100*(greska_p+greska)/2;
    float snaga=greska*p_poj+derivacija*d_poj+integracija*i_poj;
    snaga_delta=snaga_delta+snaga;
    if (snaga_delta<0){snaga_delta=0;}
    if (snaga_delta>255){snaga_delta=255;}
    analogWrite(6, int(snaga_delta));
    lct_r = millis();
}
```

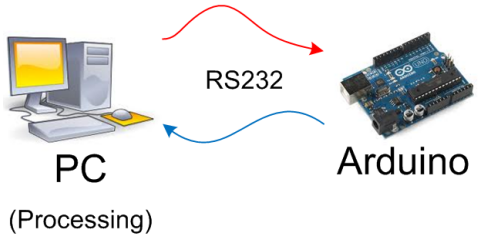
Prethodni dio koda pokazuje način na koji funkcionira PID regulator korišten u ovom diplomskom radu [2]. Na samom početku `if` naredba provjerava da li je prošlo 100 ms (vrijeme sempliranja) od posljednjeg ulaska u petlju. Nakon toga učitava se temperatura sa senzora i zaokružuje se na višekratnik 1/2 zbog greške samog senzora. Sljedeća naredba računa grešku regulacije koja je bitna za proračun izlaza PID regulatora. Sljedeći blok od 6 naredbi proračunava vrijednost potrebne snage regulatora te se tada ona pridodaje snazi iz prethodnog koraka. Nakon toga ukupna se snaga mora limitirati na snagu koju je realno moguće ostvariti, a to je snaga u granicama od 0 do 255. Na posljetku preslikavamo vrijednost potrebne snage na pin broj 6. Vrijednosti PID pojačanja dobivene su pomoću *Takahashi*-jeve metode.

2.6.5 Realizacija serijske komunikacije

U našem radu *Arduino* upotrebljava dvije serijske veze, jednu sa računalom i drugu programsku sa GPS modulom. Način primanja podataka serijskom vezom od GPS modula, ova serijska veza ne šalje podatke, opisan je u poglavlju o GPS modulu.

Serijska veza sa računalom služi za slanje mjerenih podataka i za primanje naredbi. Slijedi dio koda koji prima naredbe putem serijske veze:

```
while (Serial.available() > 0){
        inChar = Serial.read();
        inData[index] = inChar;
        index++;
```



Slika 2.12: Serijska komunikacija Arduino – PC

```
}
if (index > 2){
  if (strcmp(inData,"STOP")  == 0) {
      reg_temp=0;
  }
  else if (strcmp(inData,"TM01")==0) {
      reg_temp=20;
  }
  else if (strcmp(inData,"TM02")==0) {
      reg_temp=21;
  }
  for (int i=0;i<19;i++) {
       inData[i]=0;
  }
  index=0;
}
```

Petlja `while` nam uvjetuje ponavljanje koda sve dok serijska veza prima podatke. Kod u petlji znak po znak naredbe sprema u varijablu `inData` koja predstavlja string. Kad od računala više ništa ne primamo napuštamo petlju i provjeravamo dali smo primili sve znakove (svaka naredba sadrži točno četiri znaka), dakle ako je `index` varijabla veća od 2 primili smo sve znakove. Slijedeće su naredbe uspoređivanja npr. ako je primljena naredba `TM01` postavljamo temperaturu na 20 (`reg_temp=20;`), nakon izvršenja naredbe varijable `inData` i `index` postavljamo u nulu kako bi mogli ponoviti ciklus.

Slanje podataka na računalo sa serijskom vezom riješeno je na ovaj način:

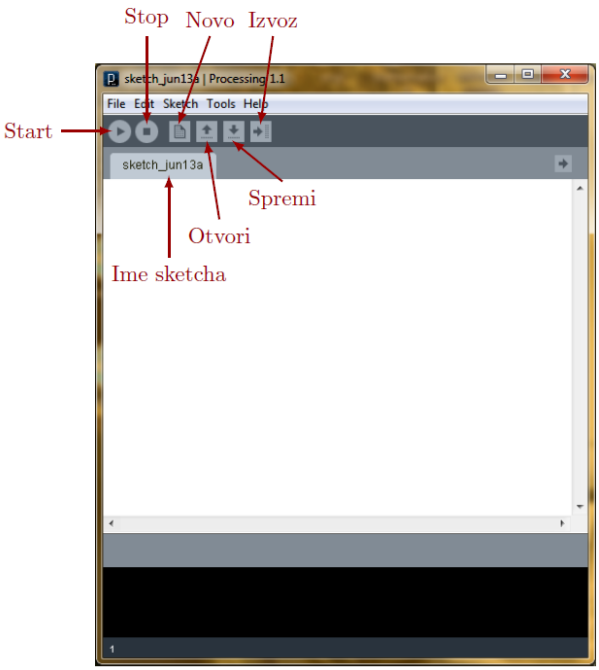
```
if(millis() - lastConnectionTime > 30*1000) {
        Serial.write("$TEMP,");
        Serial.write(temp);
        Serial.write("$TEMP1");
        Serial.write(temp1);
        lastConnectionTime = millis();
```

```
}
```

`If` naredbom provjeravamo dali je prošlo 30 sekundi od prošlog slanja podataka ako je prošlo onda podatke ponovno šaljemo. Prvo se šalje ključna riječ `$TEMP,` koja predstavlja temperaturu okoline, a tada varijabla u kojoj je spremljena temperatura okoline. Sljedeća se šalje temperatura u postavu sa ključnom riječi `$TEMP1`. Primjećujemo da svaka ključna riječ počinje sa znakom "$" što će nam kasnije u Processingu olakšati prepoznavanje početka novog podatka.

Programski paket “Processing”

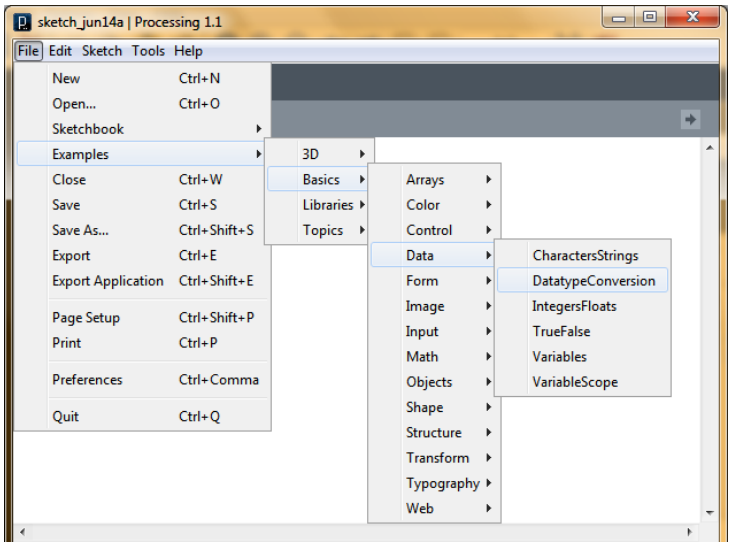
Program processing besplatan je alat dostupan na internetskoj stranici *processinga* [1] [3]. Program nije potrebno instalirati već samo pokrenuti ”.exe” datoteku koju ste skinuli i vaš rad u processingu može početi. Sučelje programa vrlo je jednostavno i izgleda kao jednostavan tekst editor. Svaki sketch [2] ima mjesto za pisanje koda, gumb za spremanje, otvaranje i za pokretanje programa (sl. 3.1).



Slika 3.1: Sučelje processinga

Da bi provjerili da vam sve radi možete pokrenuti neki gotov primjer u processingu. Gotove primjere možete naći u izborniku FILE -> EXAMPLES. Kada ste otvorili primjer kliknite na gumb “Run” ako se otvori novi prozor u kojem se izvodi primjer to znači da ste pokrenuli vaš prvi primjer u processingu. Nakon što ste provjerili dali sve radi sve je spremno da počinjete raditi vaš prvi sketch. Klikom na gumb “new” otvara vam se novi prazan sketch.

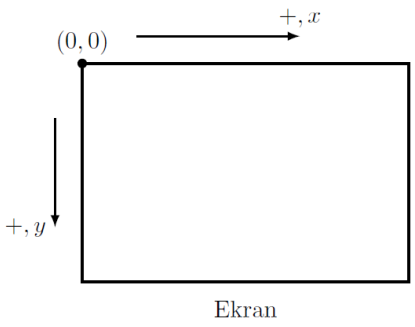
[1] http://www.processing.org/
[2] processing svaki svoj prozor naziva “sketch”



Slika 3.2: Odabir gotovih primjera

3.1 Jednostavne konture i boje

Koordinatni sustav koji koristi *processing* nije standardni kartezijev koordinatni sustav već ima ishodište u gornjem-lijevom kutu. Njegova x os raste u smjeru prema desno, a y os raste prema dolje.



Slika 3.3: Koordinatni sustav

Svaki osnovni oblik mora imati neke osnovne parametre da bi se mogla odrediti npr. njegova širina, visina, duljina, kasnije i boja itd. Najjednostavniji oblik u procesnigu je točka. Za crtanje točke potrebna su dva ulazna parametra x i y koordinata. Naredba glasi:

```
point(20,60);
```

gdje su 20 i 60, x i y koordinate točke na ekranu. Dužinu je također jednostavno nacrtat, trebaju nam dvije točke:

```
line(10,30,80,60);
```

Ova naredba crta dužinu kroz dvije točke, točku `point(10,30)` i točku `point(80,6)`. Naredba za crtanje pravokutnika je:

```
rect(20,30,50,40);
```

Međutim postoji više načina crtanja pravokutnika, a to su:

- **CORNER** pravokutnik definiran sa gornjim-lijevim kutom te širinom i visinom
- **CENTER** koordinate središta te širina i visina
- **CORNERS** koordinate gornjeg-lijevog i donjeg-desnog kuta

Način crtanja pravokutnika poziva se naredbom `rectMode()`. Nakon što smo naučili crtati pravokutnik, crtanje elipse bit će jednostavno. Zapravo crtanje elipse gotovo je identično crtanju pravokutnika. Naredba za crtanje elipse je `ellipse()`. Zadani način za crtanje elipse je "CENTER", a ne "CORNER" kao kod `rect()`. Pozivanje načina kod elipse vrši se naredbom `ellipseMode()`.

Važan element pored jednostavnih kontura je korištenje boja. Najjednostavnije boje u *processingu* su nijanse sive. Nijanse sive se označavaju sa brojevima *0-255*. I to na način da *0* znači crnu boju, a *255* znači bijelu boju. Brojevi između njih daju nijanse sive boje. Postoje tri funkcije za korištenje boja, a to su:

- `stroke()` - boja konture objekta,
- `fill()`- boja ispunjenja unutrašnjosti objekta,
- `background()` - određuje boju pozadine prozora u kojem crtate.

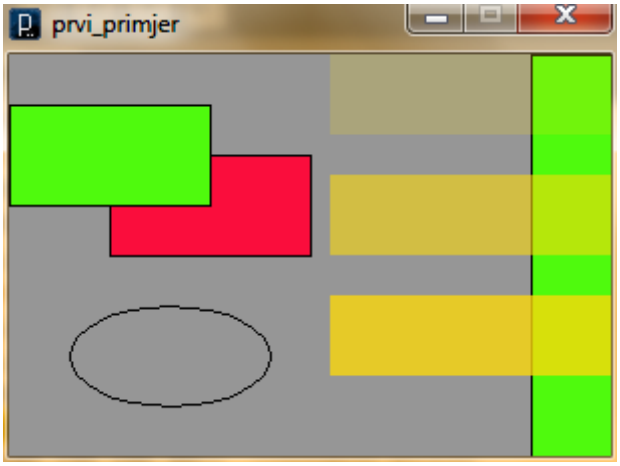
Postoje i funkcije `noStroke()` i `noFill()` koje služe za prekidanje funkcija `stroke()` i `fill()` ali je važno zapamtiti da se ne smiju koristit zajedno jer se na ekranu neće ništa vidjeti. *Processing* također omogućava korištenje RGB boja. Kod korištenja RGB boja imamo više argumenata točnije tri argumenta. Svaki od tih argumenata označava po jednu osnovnu [3] boju. Argument za svaku boju je broj od *0-255*, *0* znači da te boje nema, a *255* znači da te boje ima koliko je najviše moguće. U padajućem izborniku "Tools" postoji alat pod nazivom "Color Selector". Taj nam alat omogućuje da odabirom boje iz vizualne palete boja odaberemo boju i nijansu a alat nam vrati visinu argumenata R, G i B. Kod boja postoji i četvrti argument koji se zove "Alpha" [4], a označava transparentnost boje. On također ima vrijednost u rasponu od *0-255*, gdje *0* označava potpuno transparentnu boju, a *255* potpuno neproziran objekt.

[3] U RGB načinu označavanja boja osnovne boje su R-crvena, G-zelena, B-plava

[4] Ne mora se koristit, a kad se ne koristi vrijednost mu je *255*

Ovo je primjer kod-a korištenja naredbi za crtanje osnovnih kontura i korištenja boja u procesingu:

```
size(300,200);            //veličina ekrana
background(150);          //siva
stroke(0);                //crna
fill(250,13,60);          //crvena
rect(50,50,100,50);       //crtanje pravokutnika
rectMode(CENTER);         //način crtanja
fill(79,250,13);          //zelena
rect(50,50,100,50);
rect(280,100,40,200);
noFill();
ellipse(80,150,100,50);   //crtanje elipse
rectMode(CORNERS);
noStroke();               //nema konture
fill(252,217,10,50);      //žuta, prozirnost 80.4%
rect(160,0,300,40);
fill(252,217,10,150);     //žuta, prozirnost 41.2%
rect(160,60,300,100);
fill(252,217,10,200);     //žuta, prozirnost 21.6%
rect(160,120,300,160);
```



Slika 3.4: Prvi primjer

3.2 Funkcije setup() i draw()

Ako želimo postići da naš program postane dinamičan moramo koristiti funkcije navedene u podnaslovu. Funkcija `setup()` navodi se prva te se u njoj navode inicijalne naredbe za

naš program npr. pozadina sketcha,veličina sketcha itd. koje se učitavaju samo jedanput i to tokom pokretanja programa. Funkcija `draw()` navodi se druga te se ona izvodi stalno ponovo, najčešće 30 puta u sekundi za glatku animaciju. Broj ponavljanja u sekundi ove funkcije može se zadat pomoću naredbe `frameRate()`. Ako u funkciji `draw()` navedemo neke varijabilne koordinate [5] objekti će se pokretati po prozoru. U ovom poglavlju spominjali smo varijabilne koordinate te ćemo jednu vrstu njih sada i objasniti. To su varijable `mauseX` i `mauseY` kao što se iz imena varijabli može zaključiti to su varijable koje daju trenutne koordinate miša. Uz ove naredbe otvaraju se nebrojene nove mogućnosti npr. sad možemo crtati pravokutnike čije dimenzije ovise o poziciji miša. `pmauseX` i `pmauseY` su dodatne varijable u kojima je spremljena vrijednost koordinata miša koje su bile u prethodnom frame-u.

Ovo je primjer programa koji koristi funkcije `draw()` i `setup()` te varijabilne koordinate, koordinate miša. Program crta kontinuiranu liniju:

```
void setup() {
size(200,200);
background(255);
smooth();             // uključuje "anti-aliasing" glatki rubovi
}
void draw() {
stroke(0);
line(pmouseX ,pmouseY ,mouseX ,mouseY);    //crta liniju ovisno
                                           //o koordinatama miša
}
```

Slika 3.5: Drugi primjer

[5] Koordinate koje ovise o pomaku miša, pomicanju nekih slider-a, vremenu, itd.

3.3 Varijable

Ako neku vrijednost u svojoj aplikaciji trebamo pozvati više puta ili ako dimenzija nekog objekta ovisi o nekoj funkciji moramo koristiti varijable. U *processingu* se varijable navode prije početka pisanja programa, u zaglavlju. U zaglavlju se može navesti samo deklaracija varijable, a kasnije se varijabla može inicijalizirati. Deklaracija ima sintaksu `int varijabla;`. Deklaracija varijable je računalu potrebna da zna koliko memorije mora rezervirat za tu varijablu. Neke vrste varijabli su:

- *boolean*: logička varijabla (istina ili laž)
- *char*: slovo, 'a', 'b', 'c' itd.
- *int*: cjelobrojni broj
- *float*: decimalni broj

Inicijalizacija varijable jest zapravo punjenje adrese varijable te ima sintaksu `varijabla=50;`. A inicijalizacija i deklaracija mogu se pozvati kombinirano prema sintaksi `int varijabla=50;`. *Processing* ima neke ugrađene sistemske varijable, neke od njih smo naveli (npr. `mouseX` itd.), a ostale su:

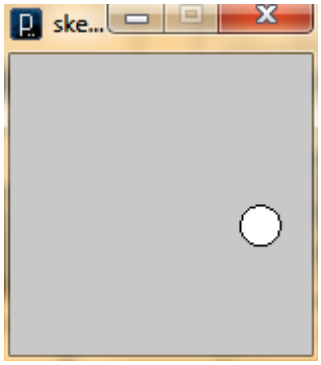
- width: širina prozora u pixelima
- height: dužina prozora u pixelima
- frameCount: broj precessiranih frame-ova
- frameRate: broj processiranih frame-ova u sekundi
- key: posljedna pritisnuta tipka na tipkovnici
- keyCode: brojčani kod pritisnute tipke
- keyPressed: Da ili ne? Jeli tipka pritisnuta?
- mousePressed: Da ili ne? Jeli tipka miša pritisnuta?
- mouseButton: Koja tipka miša je pritisnuta? Lijeva, desna ili središnja?

Ovo je primjer korištenja varijabli definiranih od strane korisnika i sistemskih varijabli. Primjer kružnog gibanja objekta po prozoru:

```
int podloga=200;  //boja podloge
float xpos=0;     //početna pozicija x
float ypos=0;     //početna pozicija y
float kut=0;      //početni kut
```

```
void setup(){
  size(150, 150);
  background(podloga);
  frameRate(30);
}

void draw(){
  background(podloga);
  xpos=width/2+sin(kut)*50;     //računanje nove pozicije
  ypos=height/2+cos(kut)*50;
  ellipse(xpos,ypos,20,20);
  kut=kut+0.05;                 //povećavanje kuta
 }
```



Slika 3.6: Treći primjer

3.4 Kontrola toka programa

Da program ne bi vrtio uvijek iste naredbe, moguće je kontrolirati tok programa nekim naredbama koje će biti opisane u ovom poglavlju. Jedna od kontrola programa je naredba `if`, ona vrti jedan određeni dio programa ovisno o nekom uvjetu. Uvjet je logičkog tipa (boolean) dakle ili ulazi u petlju ili je preskače. Sintaksa naredbe je:

```
if (uvjet){
   naredbe A;
} else {
   naredbe B;
}
```

Dakle ako je zadovoljen "uvjet" onda se nastavlja dio koda "naredbe A;", a ako "uvjet" nije zadovoljen nastavlja se dio koda "naredbe B;". Naredba se može koristit i bes `else` što je

skraćena verzija naredbe. "Uvjet" da uđemo u petlju može biti npr. da je x veće od 5 tada bi "uvjet" zapisali: `(x>5)` pri čemu je > relacija. Relacije koje se koriste u processingu su: > veće od, < manje od, >= veće ili jednako, <= manje ili jednako, == jednako i != različito. Mnogo puta nam se može dogodit da nam je ulazak u petlju uvjetovan s više uvjeta, tada uvjete možemo "spajat" s logičkim operatorima. Stoga imamo logičke operatore: (ili) ||, (i) && te (ne) ! . Logički operator ćemo koristit ako na primjer želimo ući u petlju ako je x veći od 5 i manji od 10 tada će uvjet imati oblik: `(x>5 && x<10)`. Sljedeća od naredbi za grananje programa je `while`. Petlja koja ponavlja dio koda tako dugo dok je uvjet zadovoljen. Sintaksa naredbe:

```
while (uvjet){
   naredbe;
}
```

Još jedna naredba ponavljanja `for`. Sintaksa ove naredbe je:

```
for (deklaracija i inicijalizacija, uvjet, pribrojavanje){
   naredbe;
}
```

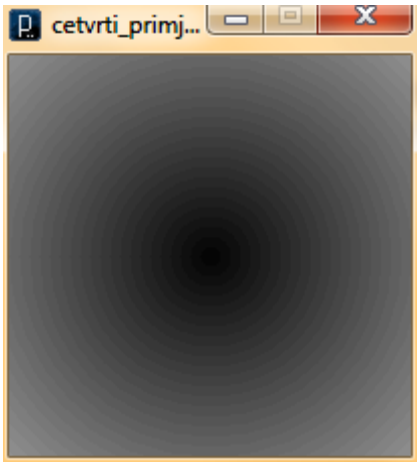
U argumentima naredbe `for` imamo tri argumenta. Prvi argument je "inicijalizacija i deklaracija" varijable o kojoj ovisi ulazak u petlju, "uvjet" ulaska u petlju, "pribrojavanje" promjena varijable o kojoj ovisi ulazak u petlju.

Ovo je primjer korištenja `for` petlje. Primjer crta koncentrične kružnice različitih nijansi sive boje:

```
int i;

void setup(){
  size(200,200);
}

void draw(){
  for (i=255; i>=0; i=i-5){
      noStroke();
      fill(i);
      ellipse(100,100,i*2,i*2);
  }
}
```



Slika 3.7: Četvrti primjer

3.5 Rad sa slikama i tekstom

Kada u processingu želimo koristiti slike moramo koristiti klasu `PImage`. Nova instanca na klasu stvara se naredbom `loadImage()`. `LoadImage()` ima jedan argument (string) koji predstavlja putanju slike koju želimo učitati. Tako učitana slika sprema se u varijablu. Kad sliku želimo nacrtati u aplikaciji to radimo naredbom `image(img,0,0,20,30)`, gdje je img varijabla u koju je spremljena prethodno učitana slika, prve dvije nule su koordinate gornjeg-lijevog kuta. Četvrti i peti argument (20,30) su koordinate donjeg-desnog kuta.

Da bi koristili font u processingu prvo ga moramo kreirati. Kreiranje fonta radimo preko pot-programa *Create Font* kojeg možemo naći u izborniku *Tools*. Tu odabiremo vrstu i veličinu fonta te pritisnemo *OK*. Program sada kreira datoteku fonta sa nastavkom *.vlw* te će je kopirati u root folder naše aplikacije.

Nakon toga potrebno je definirati objekt `Pfont`. Nakon toga možemo učitati font naredbom `loadFont()` čiji je argument ime fajla fonta koji smo kreirali. Zadnji korak je definiranje fonta teksta naredbom `textFont(f,36)` gdje je prvi argument varijabla u koju je učitan naš kreirani font, a drugi je veličina fonta. I sada možemo ispisati font naredbom `text("Ovo je tekst!",10,100)`, prvi argument je string koji ispisujemo, a druga dva su koordinate na koje tekst želimo postaviti.

3.6 Interakcija s korisnikom

Vrlo važan dio ovakvih aplikacija je interakcija s korisnikom. Osnovne funkcije za interakciju s korisnikom su `mausePressed()` klik miša te otpuštanje miša `maouseRelased()` (pr. 5). Analogne funkcije ali za tipkovnicu su `keyPressed()` i `keyReleased()`. Postoje još

dvije funkcije koje se pozivaju tjekom gibanja miša prva je `mouseMoved()` poziva se svaki put kad se miš giba i tipka na njemu nije pritisnuta. Druga `mouseDragged()` poziva se svaki put kad se miš giba i tipka na njemu je pritisnuta. Osim ugrađenih funkcija koje smo nabrojili postoje funkcije koje se mogu učitati iz gotovih modula. U diplomskom radu korišten je modul controlP5 koji sadržava gotove slider-e i radial button-e.

3.7 Rad sa mrežom

Za rad sa mrežom u processingu se koristi dodatak `net`. Dodatak `net` čini čitanje i slanje podataka između računala na mreži vrlo jednostavnim. Dodatak ima dvije klase `Client` i `Server`. Klasa `Client` se koristi za kreiranje klijenta koji se spaja na server i razmjenjuje podatke. Klasa `Server` se koristi za kreiranje servera koji šalje i prima podatke od klijenata. Također je u ovom diplomskom radu korišten dodatak, `proXML`, za pojednostavljeno korištenje SOAP/XML komunikacijskog protokola.

3.7.1 Klasa Client

Klasa `Client` se koristi za kreiranje klijenta koji se spaja na server i te s njim razmjenjuje podatke. Klijent se kreira naredbom: `myClient = new Client(this, "127.0.0.1", 5240)` Gdje je drugi argument IP adresa servera, a treći argument port na koji se želimo spojiti. Klasa `Client` posjeduje više metoda, a najvažnije su `write()` pomoću koje se može poslati upit na server, `readString()` čita string iz privremene memorije klijenta, `available()` vraća broja bajtova u privremenoj memorije koji čekaju da ih se pročita.

Ovo je primjer korištenja `Client` klase. Klijent se spaja na server "arduino-matija.dnsdynamic.com" kroz port `6883` te dohvaća datoteku "get_ip.php". Nakon toga čeka odgovor i kad ga primi ispisuje ga u konzolu naredbom `println(data);`:

```
import processing.net.*;

Client c;
String data;

void setup() {
  size(200, 200);
  background(50);
  fill(200);
  c = new Client(this, "arduino-matija.dnsdynamic.com", 6883);
```

```
  c.write("GET /ajax/get_ip.php HTTP/1.1\n");
  c.write("Host: arduino.cc\n\n");
}

void draw() {
  if (c.available() > 0) {
    data = c.readString();
    println(data);
  }
}
```

3.7.2 Klasa Server

Klasa `Server` omogućava slanje i primanje podataka pema klijentu. Kada pokrenemo server on počinje slušati port koji smo mu zadali kao argument. Pokretanje servera je vrlo jednostavno: `myServer = new Server(this, 5204);`

gdje je drugi argument port koji želimo slušati. Važne metode `Server` klase su `avaliable()` prelazi na novog klijenta, `server.readString()` čita vrijednost klijenta, `write()` šalje vrijednost svim spojenim klijentima.

Ovo je primjer korištenja `Server` klase. Pokrećemo server te slušamo port 5224. Kontinuirano provjeravamo da li je klijent spojen. Kad je klijent spojen provjeravamo dali je nešto poslao te ako je nešto poslao onda to ispisujemo u konzolu:

```
import processing.net.*;
Server myServer;

void setup()
{
  myServer = new Server(this, 5224);
}

void draw()
{
  Client thisClient = myServer.available();
  if (thisClient !=null) {
    String whatClientSaid = thisClient.readString();
    if (whatClientSaid != null) {
      println(thisClient.ip() + "t" + whatClientSaid);
    }
```

```
  }
}
```

3.7.3 Dodatak proXML

Dodatak `proXML` pojednostavljuje processingu čitanje i pisanje XML datoteka [4]. Također omogućuje hvatanje datoteka putem interneta. Iz tog razloga je vrlo koristan jer je XML struktura podataka postala standard za razmjenu podataka putem web-a. Svaki XML tag ovom dodatku predstavlja čvor zvan i element. Čvor može biti tekstualni. Ako je čvor XML element može imati atribute. Čvor koji se nalazi u nekom drugom čvoru naziva se dijete tog elementa.

Opisanu strukturu podataka koristi i SOAP [6] komunikacijski protokol. Protokol je neovisan o vrsti platforme, a koristi se za razmjenu podataka preko HTTP protokola. Razvijen je kako bi se omogućila jednostavna komunikacija tekstualnim sadržajem. Protokol je neovisan o programskom jeziku, platformi i jednostavno proširiv.

Izgled odgovora (primanje IP adrese od servera)

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <ip_address>
    <ip>192.168.100.252</ip>
  </ip_address>
</soap:Envelope>
```

Primjer čitanja IP adrese iz XML datoteke u gornjem primjeru (Izgled SOAP odgovora). Program čita IP adresu i ispisuje je u konzoli.

```
import proxml.*;
proxml.XMLElement ip_address;
XMLInOut xmlInOut;

void setup() {
   xmlInOut = new XMLInOut(this);
   xmlInOut.loadElement("get_ip.xml");
}

void xmlEvent(proxml.XMLElement element){
  ip_address = element;
```

[6] Simple Object Access Protokol

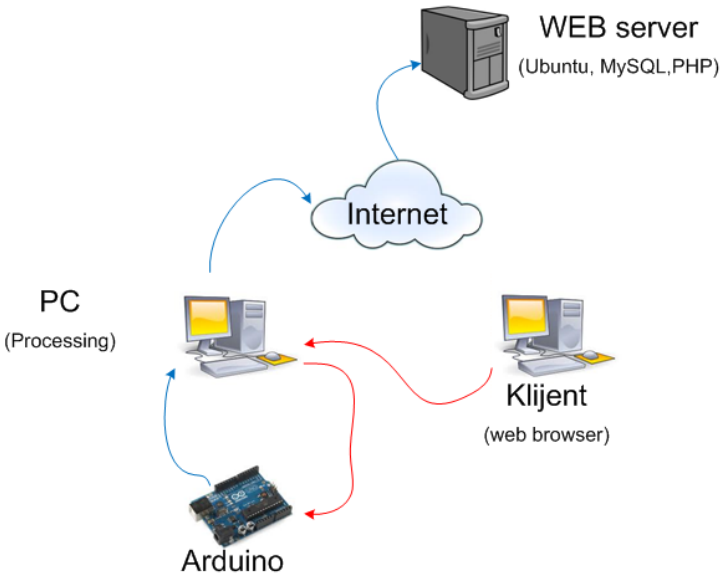
```
println(ip_address.getChild(0).getChild(0).getChild(0).toString());
}
```

3.8 Rad sa serijskom vezom

Uspostavljanje veze između računala i arduina vrši se serijskom vezom. Kako bi serijska veza radila u proceesing aplikaciji postoji klasa `serial`. Klasa `serial` omogućava slanje i primanje podataka putem serijske veze. Serijski port se otvara pomoću naredbe: `Serial(this, COM1, 9600, N, 8, 1.0)` Gdje je `COM1` ime porta koji otvaramo, `9600` je brzina komunikacije, `N` paritet podatka (može biti "*E*" paran, "*O*" neparan), *8* broj bitova u podatku te *1.0* broj stop bitova.

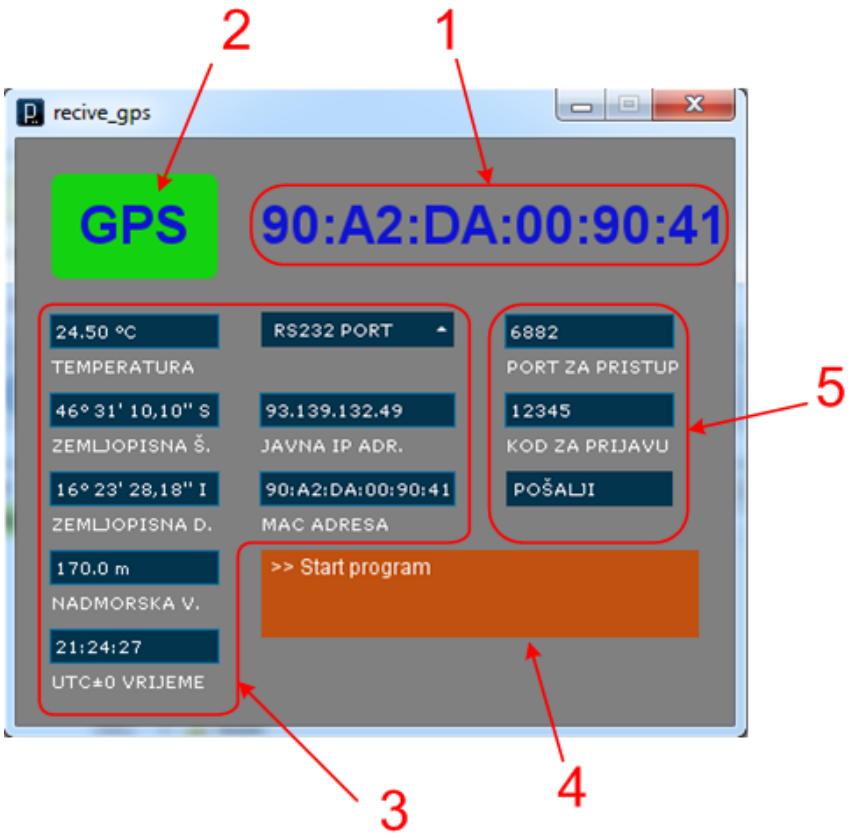
Bitne metode klase serial su `stop()` zatvaranje porta, `available()` vraća broj dostupnih bajtova, `read()` čita dostupne podatke, `readString()` čita podatke i vraća ih kao string tip podataka, `write()` šalje podatke te `list()` vraća listu dostupnih serijskih portova.

3.9 Realizacija aplikacije



Slika 3.8: Struktura komunikacije - Processing

Processing aplikacija u ovom diplomskom radu služi kao veza između krajnjeg klijenta, servera te Arduina (sl. 3.8). Aplikacija prima podatke od Arduina putem serijske veze te ih prosljeđuje na web server putem etherneta (sl. 3.8 - plava linija). Isto tako putem etherneta prima podatke te ih obrađuje i obliku naredbi, za upravljanje regulacijom, ih šalje na Arduino putem serijske veze (sl. 3.8 - crvena linija).



Slika 3.9: Aplikacija za primanje i slanje informacija

Sučelje aplikacije podijeljeno je na nekoliko dijelova:

- **(1)** MAC adresa Arduino uređaja s kojim je računalo povezano. Ova je adresa vrlo važna kod slanja podataka na web server na osnovu nje web server zna kamo smjestiti primljene podatke.
- **(2)** indikator rada GPS sustava , ako GPS senzor vidi dovoljno satelita za rekonstrukciju pozicije onda je indikator zelene boje inače je crvene.
- **(3)** dio predviđen za ispis primljenih podataka od Arduina i web servera.
- **(4)** "message box" u koji se upisuju poruke o radu pozadinskih akcija (slanje, primanje podataka. . .).
- **(5)** podaci koje upisujemo ručno i šaljemo na server pritiskom na dugme "pošalji".

Aplikacija podatke sa Arduino uređaja prima putem serijske veze te ih ispisuje na ekran i prosljeđuje na web server. Podaci koje apikacija šalje na web server su mjereni podaci i još globalna IP adresa računala te MAC adresa Arduina. Globalna IP adresa računala važna je kako bi klijent znao na koju adresu poslati naredbe. MAC adresa Arduina također je vrlo važna kako bi web server kasnije mogao razlučiti podatke. Podatke "port za pristup" i "kod za prijavu" možemo upisati ručno i dugmetom "pošalji" poslat ih na web server. Aplikacija ima mogućnost

promjene porta za pristup iz razloga da se zadani port za pristup koristi u nekom drugom servisu. Kod za prijavu služi kako bi na web serveru uređaj povezali sa korisnikom web sučelja. Ako ćemo u web sučelju znati kod za prijavu za uređaj dotične MAC adrese sustav je siguran da je taj uređaj naš iz razloga jer smo sami u processingu upisali kod za pristup. Na taj način ćemo u web sučelju postati vlasnik uređaja.

Zbog velike količine programskog koda koji je bio potreban za realizaciju processing aplikacije kod će biti priložen na CD-u uz diplomski rad. Svi dodaci koji su korišteni u programiranju aplikacije opisani su prethodnim poglavljima.

Izrada WEB sučelja

Web sučelje putem kojeg se upravlja Arduino uređajima sa klijentske strane izrađeno je u obliku HTML stranice. HTML stranica je dinamička te se dinamički kreira kombinacijom PHP/MySQL programskih jezika.

4.1 PHP

PHP [1] je objektno-orijentiran programski jezik namijenjen prvenstveno programiranju dinamičkih web stranica [5]. Drugim riječima PHP je skriptni programski jezik pomoću kojeg je moguće kreirati HTML stranicu popunjenu dinamičkim sadržajem na samom poslužitelju prije nego što je ona poslana klijentu. Ovim načinom generiranja sadržaja klijent ne može vidjeti kod (skriptu) koji je generirao sadržaj koji gleda, već ima pristup čistom HTML kodu. Važno je napomenuti da je PHP "open-source" programski jezik što znači da svatko tko želi može koristiti izvorne PHP kodove pisane u C programskom jeziku i iste, ukoliko ih razumije, mijenjati po svojoj volji te na taj način dodavati nove funkcije PHP-u. Iz spomenute definicije PHP-a lako je zaključiti kako se programiranjem u PHP-u zapravo dobiva takozvani poslužiteljski kod koji se isključivo izvršava na poslužitelju, a ne na korisničkom računalu koje zapravo dobiva samo gotov HTML kod koji se prikazuje kroz internetski preglednik. Server-side skripte se izvršavaju na poslužitelju nakon što isti primi zahtjev za PHP dokumentom. Nakon primitka zahtijeva za PHP dokumentom poslužitelj izvršava PHP kod i na osnovu njega generira HTML kod koji šalje klijentu. PHP je jedna od najnaprednijih i najkorištenijih server-side skriptnih tehnologija koje su danas u upotrebi. Još jedna važna stvar je činjenica da je PHP bogat funkcijama za manipuliranje velikim brojem različitih tipova sadržaja. Od manipuliranja grafikom (png, jpg, flash...) do učitavanja .NET modula i rada sa XML-om. Osim toga PHP nudi i podršku za baratanje širokom paletom baza podataka. Podržava sve popularnije baze podataka kao što su MySQL, PostgreSQL, dBase, Oracle, ODBC itd.

[1] HyperText Preprocessor

4.2 MySQL

Prije upuštanja u rad sa MySQL-om potrebno je odrediti odgovarajući izgled baze podataka, odnosno napraviti shemu baze, koja se u kasnijem postupku prevodi u određen broj tablica koje se koriste za pohranjivanje podataka [6]. Osnovni element koji se pohranjuje u bazi naziva se entitet, entitet može biti bilo što: osoba, neki objekt, događaj, služba u nekoj organizaciji i sl. dakle stvari iz stvarnog života o kojima želimo čuvati informacije. Drugi važan pojam u teoriji baza podataka je relacija. Kao što u stvarnom životu postoje određeni međusobni odnosi između dvije ili više osoba, događaja i sl. tako se u bazama podataka mogu pojaviti određeni odnosi ili relacije između raznih entiteta, koji se odgovarajući način predstavljaju unutar baze. Prema vrsti, relacije se mogu podijeliti na relacije jedan prema jedan, jedan prema više odnosno više prema jedan te više prema više. MySQL sve podatke pohranjuje unutar tablica koje se sastoje od kolona i redova. Kolone se nazivaju još i poljima ili atributima, a služe za skladištenje pojedinih podataka o određenom entitetu, redovi se nazivaju još zapisima ili slogovima i sadrže sve podatke jednog entiteta.

4.3 Realizacija web sučelja

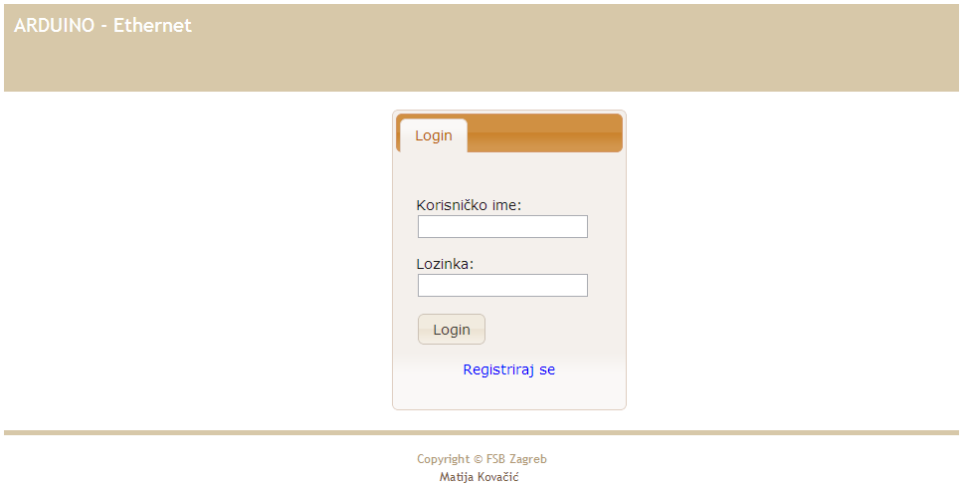
Web sučelje izrađeno za potrebe ovog diplomskog rada omogućuje krajnjem korisniku da pregledava mjerne podatke spremljene u bazu podataka. Sučelje također omogućava korisniku da djeluje na Arduino uređaj na način da putem Ajaxa [2] šalje naredbe Arduinu. Sučelje je izrađeno vrlo interaktivno te svakom, iako i nema mjerni uređaj, omogućava da se prijavi u sustav. Prijavom u sustav korisnik dobiva mogućnost da unese svoje uređaje ili da prati tuđe uređaje ako mu to drugi vlasnici uređaja dopuste. Svaki korisnik može svoj uređaj dijeliti sa drugim korisnicima. Dijeljenje uređaja sa drugim korisnicima omogućava tim korisnicima da pregledavaju mjerne podatke uređaja, vide osnovne informacije o uređaju te također mogu djelovati na uređaj. Sučelje je napravljeno tako da je pronalaženje mjernih stanica vrlo jednostavno i moguće na tri načina. Prvi način je pronalaženje stanice na vremenskoj osi, nalazi se u vremenskoj točki unosa stanice u sustav drugi je način pronalaženje na geografskoj mapi te treći način filtriranje stanica na temelju korisničkog imena korisnika koji je dodao mjernu stanicu. Nakon što stanicu nađete bilo na vremenskoj osi ili na geografskoj mapi klikom na nju otvaraju se svi podaci o vašoj mjernoj stanici. U sučelju je također omogućeno editiranje podataka mjerne stanice koju ste vi dodali i pregled svih stanica koje su vam povjerene da ih pratite.

Zbog velike količine programskog koda bit će u nastavku opisan samo način rada web sučelja, a programski kod bit će priložen na CD-u uz diplomski rad.

[2] Asynchronous JavaScript and XML

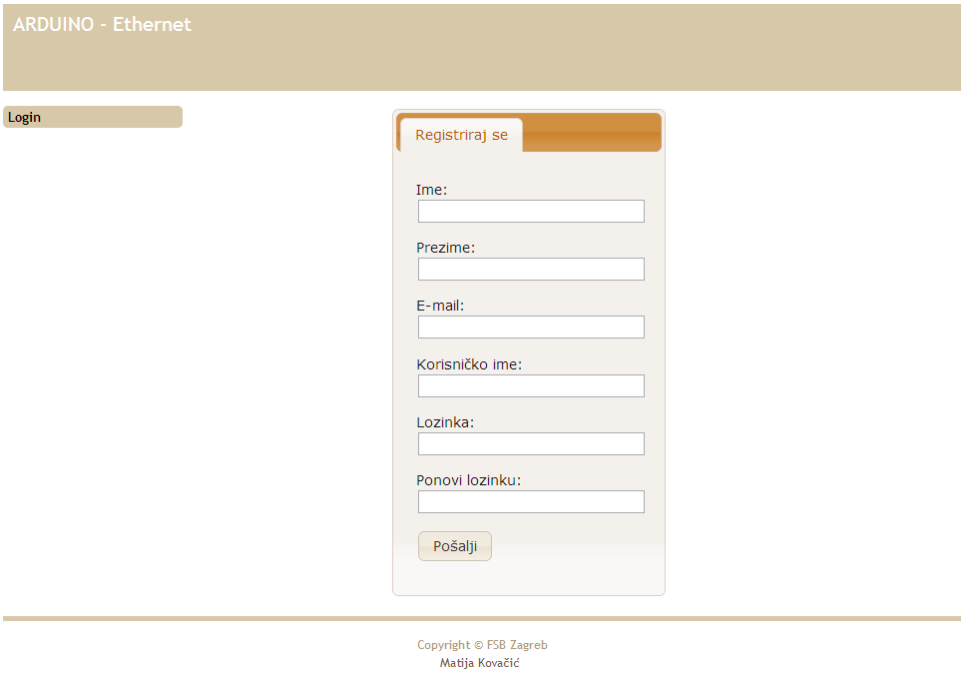
4.3.1 Prijava/registracija

Otvaranjem web sučelja otvara nam se "Login" obrazac u koji možemo upisati svoje korisničko ime i lozinku te na taj način ući u okolinu web sučelja.



Slika 4.1: Login obrazac

U slučaju da nemamo otvoren korisnički račun jednostavno ga možemo otvoriti klikom na "Registriraj se" link ispod "Login" dugmeta (sl. 4.1).



Slika 4.2: Obrazac za registraciju

U obrazac za registraciju (sl. 4.2) upisujemo ime i prezime te e-mail. E-mail mora biti točno upisan jer ćete na e-mail dobiti poruku za aktivaciju računa. Također je potrebno upisati ko-

risničko ime i lozinku. Prilikom upisa korisničkog imena sustav provjerava dostupnost tog korisničkog imena ako ono nije dostupno sustav neće dopustiti registraciju. Lozinku je potrebno upisati dva puta zbog provjere da prvi put niste pogriješili u pisanju. Pritiskom na dugme pošalji poslali ste svoju registraciju u sustav. Nakon nekoliko trenutaka primit ćete e-mail poruku u kojoj trebate slijediti link za aktivaciju računa. Nakon pritiska na link za aktivaciju otvara se početna stranica (sl. 4.3) sustava otvorena sa vašim računom.

Slika 4.3: Početna stranica

Na početnoj stranici s lijeve strane nalazi se izbornik (sl. 4.3 - 1), te gore desno korisnikovo korisničko ime i link za "logout" (sl. 4.3 - 2). Otvaranjem linka "Diplomski rad" iz izbornika s lijeve strane otvara se HTML verzija diplomskog rada.

4.3.2 Moji uređaji

Otvaranjem linka "Moji uređaji" iz izbornika otvara se stranica na kojoj možemo vidjeti popis uređaja koje smo mi dodali te nudi mogućnost dodavanja novih uređaja.

Na podstranici moji uređaji (sl. 4.4) u gornjem djelu nalazi se popis svih uređaja koje smo dodali u sustav. Klikom na svaki od njih otvaraju se osnovne informacije o uređaju. Tu možemo vidjeti opis uređaja, ime uređaja, oznaku uređaja, tko prati naš uređaja, IP adresu uređaja i port uređaja. Svaki uređaj moguće je uređivati klikom na link "Uredi" ili obrisati klikom na link "Obriši uređaj".

U donjem djelu podstranice nalazi se obrazac za unos novih uređaja. Sustav omogućava unos virtualnih uređaja, u svrhe prezentiranja rada sustava sa većim brojem uređaja, jer posjedujemo samo jedan Arduino uređaj. Uređaji se unose na način da se klika po google mapi, svaki klik na mapu generira novi uređaj koji još ne ulazi u data bazu. Prija ulaska u data bazu postoji mogućnost da se još promjene osnovne informacije o uređaju. Kad ste zadovoljni s generiranim uređajima možete ih unijeti u data bazu pritiskom na dugme "Dodaj". Ako niste

Moji uređaji (4)

mjerna stanica (Matija)

[Obriši uređaj] [Uredi]

ip za pristup uređaju:	93.139.168.224
port za pristup uređaju:	6882
korisnici koji prate uređaj:	mario, matija
Oznaka:	Narančasta
Opis:	Ovo je uređaj za mjerenje temperature u mojoj sobi. :)...

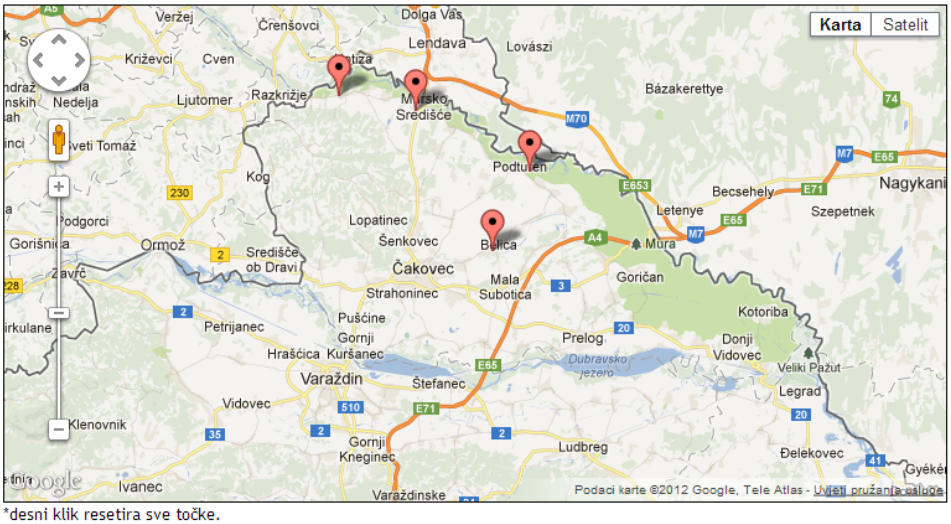
mj_st_ms

mj_st_ck

mj_sta_podturen

Dodaj uređaje

Odaberite poziciju uređaja[Zapamti poziciju karte]:



*desni klik resetira sve točke.

Uređaji

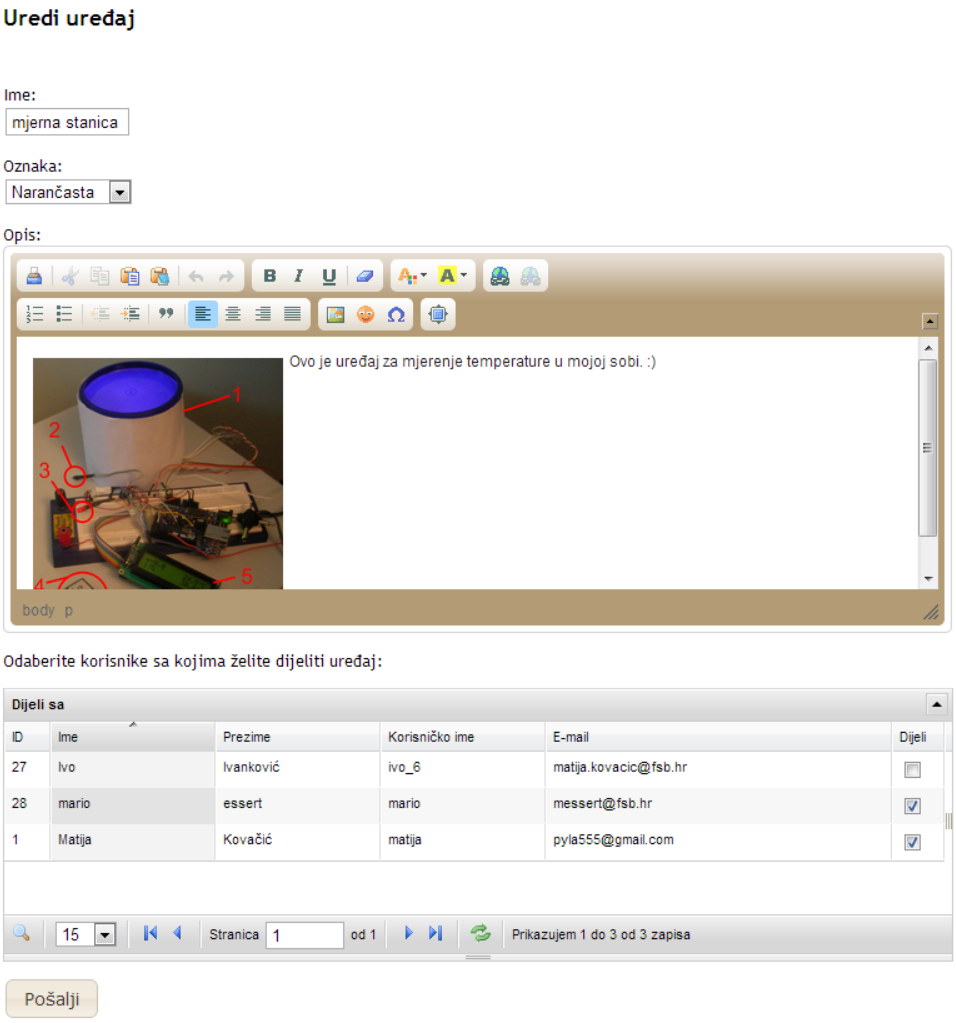
Ime:	Z. š.:	Z. d.:	Nad. visina:	Datum:	Oznaka:
mj_stanica_1	46.521266	16.344481	160	08.11.2012.	Crvena
mj_stanica_2	46.509925	16.429625	160	08.11.2012.	Crvena
mj_stanica_3	46.463594	16.555967	160	08.11.2012.	Crvena
mj_stanica_4	46.40302	16.514769	160	08.11.2012.	Crvena

Dodaj

Slika 4.4: Podstranica moji uređaji

zadovoljni generiranim uređajima možete ih obrisati desnim klikom na mapu. Kada uređaje dodajete uvijek na približno istom dijelu mape možete zapamtiti poziciju mape pritiskom na link "Zapamti poziciju karte" i sljedeći put kad otvorite ovu podstranicu mapa će se otvoriti na toj poziciji.

4.3.3 Uređivanje uređaja



Slika 4.5: Podstranica uređivanje uređaja

Kod uređivanja uređaja moguće je mijenjati ime uređaja, oznaku uređaja, opis uređaja te mijenjati korisnike sa kojima dijelite svoj uređaj. Za uređivanje opisa uređaja koristi se CKeditor [3]. CKeditor je vrlo napredan JavaScript uređivač teksta otvorenog koda sa mnogim naprednim mogućnostima. Sa njime u tekst možemo dodati sliku, video, tablicu također možemo formatirati tekst podebljanjem, podcrtavanjem, bojanjem i sl. Ispod uređivanja opisa nalazi se flexigrid [4] tablica u kojoj se nalazi popis svih korisnika sustava. Označavanjem checkbox-a pored nekog

[3] http://ckeditor.com/
[4] http://flexigrid.info/

od korisnika omogućavamo tom korisniku praćenje dotičnog uređaja i obrnuto. Flexigrid je vrlo fleksibilna JavaScript tablica otvorenog koda koja pruža mogućnost očitanja podataka putem Ajaxa. Ima vrlo napredne mogućnosti poput sortiranja zapisa, dodavanja zapisa, brisanje zapisa, traženje zapisa i sl. te je vrlo jednostavna za korištenje.

4.3.4 Uređaji koje pratim

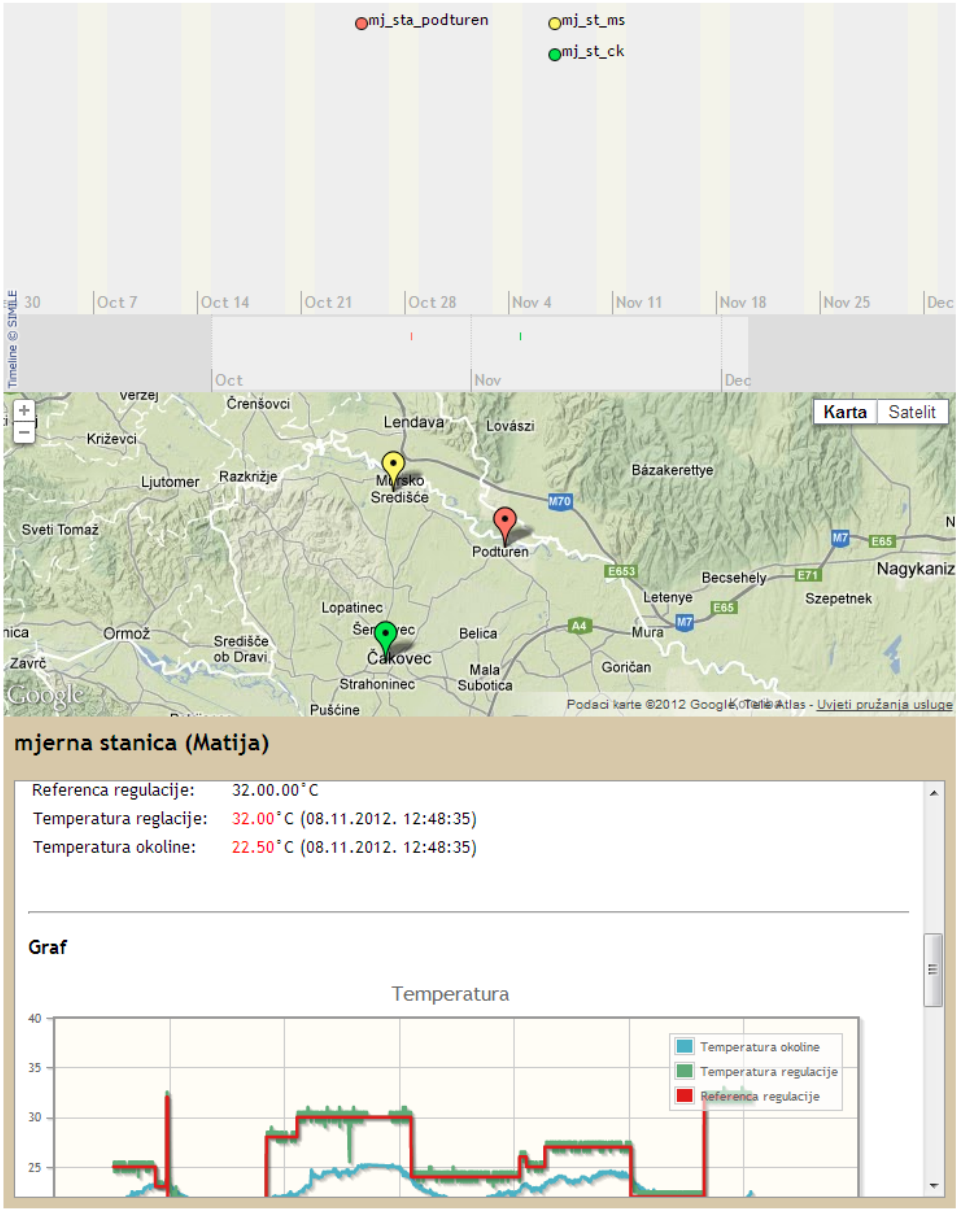
Popis uređaja koje neki korisnik prati može vidjeti na podstranici uređaji koje pratim (sl. 4.6). Klikom na ime pojedinog uređaja iz popisa otvaraju se osnovne informacije o uređaju sa popisom korisnika koji sa vama prate uređaj.

Slika 4.6: Podstranica uređaja koje pratim

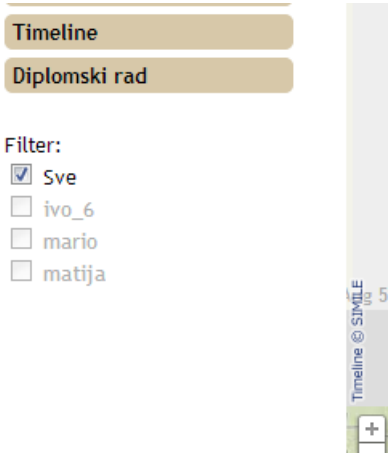
4.3.5 Timeline

Podstranica timeline (sl. 4.7) pruža nam interaktivan prikaz uređaja prostorno i vremenski raspoređenih u prostorno-vremensku os. Prostorno-vremenska os dio je MIT Simile projekta. Naše mjerne uređaje možemo naći na vremenskoj osi pomicanjem same osi uz pomoć miša. Pomicanjem vremenske osi pomičemo se u vremenu i samo uređaji koji se nalaze u periodu u kojem se nalazimo prikazuju se na prostornoj osi/mapi. Slijedeća je geografska mapa na njoj također možemo pronaći naše mjerne uređaje. Mjerni uređaj na osima vidimo kao pribadaču/mali kružić te klikom na oznaku otvaramo sve podatke o dotičnom uređaju. Podaci se otvaraju u trećem dijelu podstranice, učitavaju se pomoću Ajaxa. Ukoliko kliknete na oznaku uređaja za koji nemate dozvolu za praćenje prikazat će vam se obavijest o tome i podaci se neće učitati.

Uređaje na prostorno-vremenskoj osi moguće je filtrirati po vlasniku uređaja uz pomoć filtra (sl. 4.8) koji se nalazi ispod izbornika. Zadano je da je filtar prikazuje sve podatke ali se taj odabir može poništiti i tada se mogu odabrati vlasnici čije uređaje želite vidjeti.

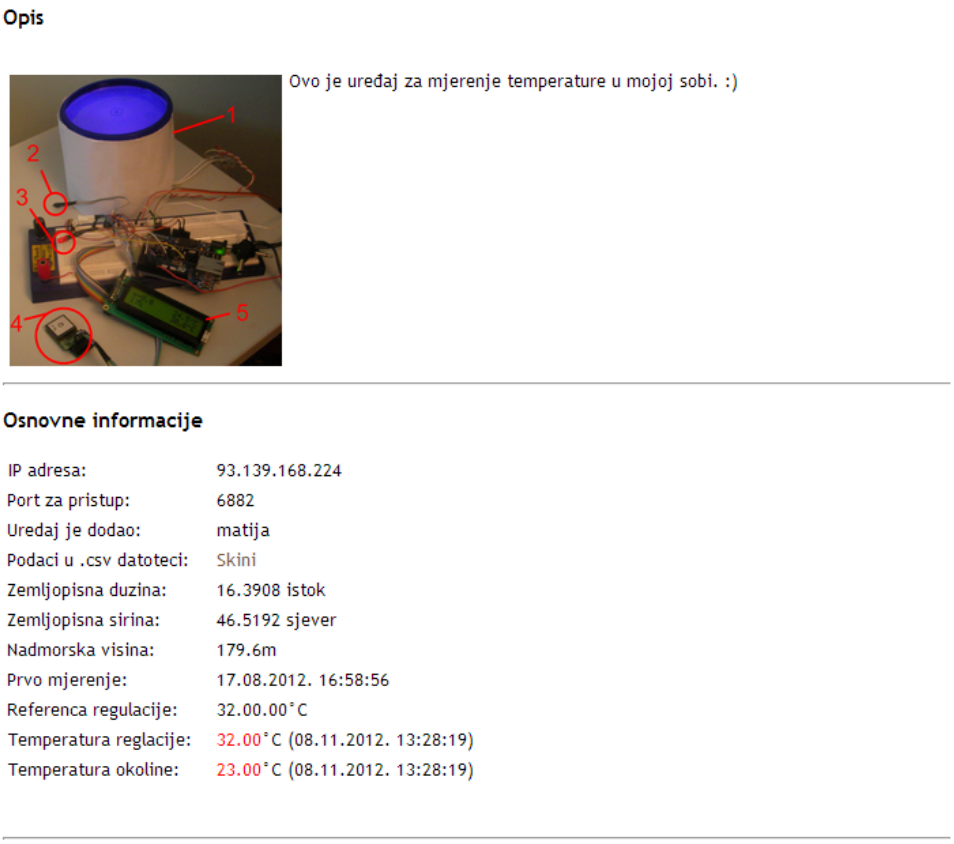


Slika 4.7: Podstranica timeline



Slika 4.8: Filtar

Podaci o uređaju koji nam se prikazuju podijeljeni su u četiri dijela. Prva dva dijela (sl. 4.9) prikazuju opis uređaja koji smo uredili u web sučelju i proširene osnovne informacije poput geografskih koordinata, nadmorske visine, vlasnika uređaja, vrijeme prvog mjerenja, trenutne temperature regulacije i okoline te link za .csv datoteku sa svim izmjerenim podacima uređaja.



Slika 4.9: Timeline, opis i osnovne informacije uređaja

Druga dva dijela (sl. 4.10) prikazuju graf izmjerenih podataka te obrazac za djelovanje na uređaj. Graf prikazuje tri grafa temperaturu okoline, temperaturu regulacije te referentnu temperaturu regulacije. Za crtanje grafa korišten je JavaScript dodatak jqPlot [5]. JqPlot je vrlo zgodan alat jer podržava način rada sa Ajax-om te omogućuje povećavanje dijela grafa sa označavanjem prostora unutar grafa. Resetiranje povećanja postiže se duplim klikom na graf. Zbog velike količine podataka bilo je potrebno segmentirati graf jer bi inače iscrtavanje grafa trajalo predugo. Segmentiranje je riješeno na način da se prvotno iscrta graf za vremensko razdoblje prethodna tri dana. Kako bi vidjeli podatke od prije tjedan dana možemo graf ponovno nacrtati uz odabir vremenskog perioda koji želimo crtati. Period se odabire na slideru ispod grafa. Slider ima dvije granice lijeva granica predstavlja vrijeme početka, a desna vrijeme kraja crtanja. Datum koji je odabrani ispisuje se iznad slidera sa narančastom bojom fonta. Odabrani period obuhvaća granične dane. Optimalni odabir razdoblja je tri dana ali se dopušta i odabir

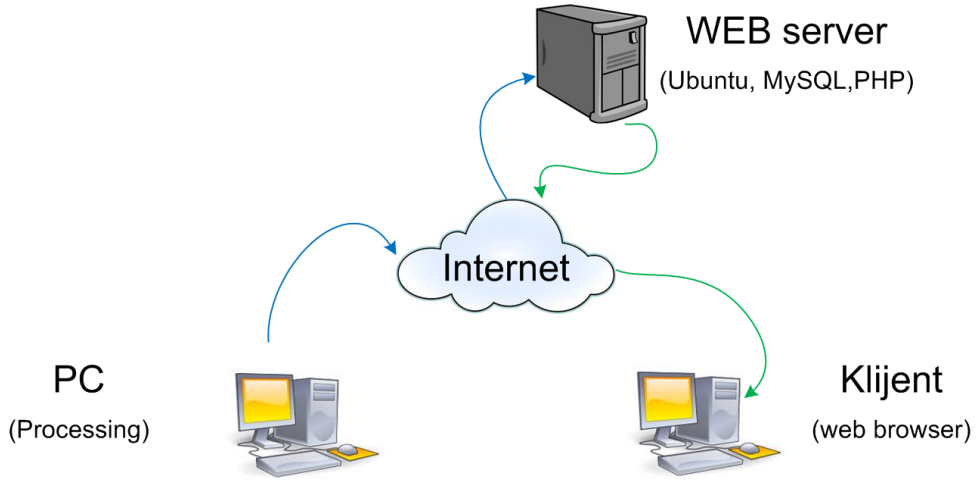
[5] http://www.jqplot.com/

dužeg razdoblja. Graf sa novim granicama iscrtava se klikom na dugme “Crtaj”. Obrazac za djelovanje na uređaj nudi nam odabir temperature iz padajućeg izbornika. Odabrana temperatura šalje se putem Ajaxa prema računalu na koji je priključen naš mjerni uređaj. IP adresu i port računala kojem šalje temperaturu sustav pronalazi u svojoj data bazi. Nakon primitka željene temperature računalo prosljeđuje željenu temperaturu serijskom vezom na Arduino. Osim odabira temperature moguće je i isključivanje i uključivanje regulacije.

Slika 4.10: Timeline, graf i postavljanje temperature regulacije

WEB server

Web server nam je u našem sustavu potreban iz dva bitna razloga. Prvi je razlog pokretanje web stranice koju smo izradili za jednostavniji i intuitivniji pristup podacima (sl. 5.1 – zelena linija), a drugi je razlog spremanje mjernih podataka (sl. 5.1 – plava linija).



Slika 5.1: Komunikacija sa web-serverom

Za operativni sustav web servera odabran je Ubuntu Server Edition [7] zbog svoje jednostavnosti, kvalitete, podrške i rasprostranjenosti također i zato što je besplatan. U osnovnoj instalaciji Ubuntu podržava PHP programski jezik i MySQL bazu podataka što će nam omogućiti izradu dinamičkog HTML sučelja i spremanje velike količine podataka dobivenih mjerenjem. U nastavku slijedi nešto o osnovama Ubuntu servera, o instalaciji servera te podešavanjima potrebnim za normalan rad našeg sustava.

5.1 Ubuntu

Ubuntu je afrička riječ koja bi otprilike upućivala na “humanost prema drugima” i potiče iz Zulu i Xhosa afričkog jezika [8]. Ako se naziv poveže sa činjenicom da je to Linux distribucija utemeljena na Debian distribuciji, dakle besplatna na zadovoljstvo svih koji je žele koristiti,

od koje je naslijedila sve njezine kvalitete, nije ni čudo da je Ubuntu i njene inačice sve popularnija distribucija iako je predstavljena 2004. godine (Debian se razvija od 1997. godine). Debian distribucija zasnovana je na potpuno volonterskom radu pripadnika FSF udruge (The Free Software Foundation), kojima je osnovni cilj razvoj besplatne programske potpore za bilo koga, u kojem trenutno u razvoju ove vrste programske potpore sudjeluje više od 1000 osoba. Ubuntu je drugačiji projekt od Debiana, ali u oba projekta uključene su osobe iz oba tima, te mu je stoga vrlo sličan u bitnim osobitostima:

- Ubuntu koristi isti paketni mehanizam nadogradnje kao Debian.
- Svakih 6 mjeseci izdaje se nova verzija operativnog sustava koja se podržava 18 mjeseci glede sigurnosnih ispravaka i programskih zakrpi za kritične greške.
- Verzije se označavaju po načelu G.MM (jedan ili dva zadnja broja Godine objave.Mjesec u Momentu objave verzije). U osnovi verzije OS izdaju se u travanju i listopadu.
- Podržano je nešto manje platformi u odnosu na Debian; i386, AMD, powerpc, sparc, ia64 i hppa.
- Otklanjanje grešaka je uzajamno jer mnogi dijelovi Ubuntu-a nisu ništa drugo do dio najnovije verzije Debian-a i obratno.
- Raspoložive su i poslužiteljske verzije ovog operativnog sustava (Ubuntu Server).

Raspoloživo je više distribucija koje se razlikuju po grafičkom sučelju (GNOME i KDE), poslužiteljske (Ubuntu Server Edition) i korisničke verzije (Ubuntu Desktop Edition), edukativna verzija (Edubuntu), osnovna minimalna u svakom pogledu besplatna verzija (Gobuntu), verzija za mobilne Internet uređaje (Ubuntu MID), verzija za slabije računalne platforme (Xubuntu) i verzija optimizirana za multimedijalne sadržaje (Ubuntu Studio). Za potrebe diplomskog rada instaliran je Ubuntu Server Edition 12.04. Server Edition pruža zajedničku bazu za sve vrste poslužiteljskih aplikacija. To je minimalistički dizajn, koji pruža platformu za željene usluge, kao što su file /print usluge, web hosting, e-mail hosting, itd. Postoji nekoliko razlika između izdanja Ubuntu Server Edition i Ubuntu Desktop. Treba napomenuti da oba izdanja koriste iste apt repozitorije, čineći jednako lakim instalirati server aplikacije na Desktop Edition kao što je na Server Edition. Razlike između dva izdanja su nedostatak X Window okoliša u Server Edition, instalacija, te različite opcije kernela.

Kernel Razlike:

- Server Edition koristi Deadline I / O scheduler umjesto CFQ schedulera, kojeg koristi Desktop Edition.
- Preemption je isključen u Server Edition.
- Timer interrupt je 100 Hz u Server Edition i 250 Hz u Desktop Edition.

5.2 Instalacija

Ubuntu 12.04 Server Edition podržava tri glavne arhitekture: Intel x86, AMD64 i ARM. Preporučeni minimalni zahtjevi za instalaciju distribucije:

Tablica 5.1: Minimalni tehnički zahtjevi "Ubuntu Server Edition 12.04"

Tip instalcije	CPU	RAM	Tvrdi disk
Server	300 MHz	128 MB	1 GB

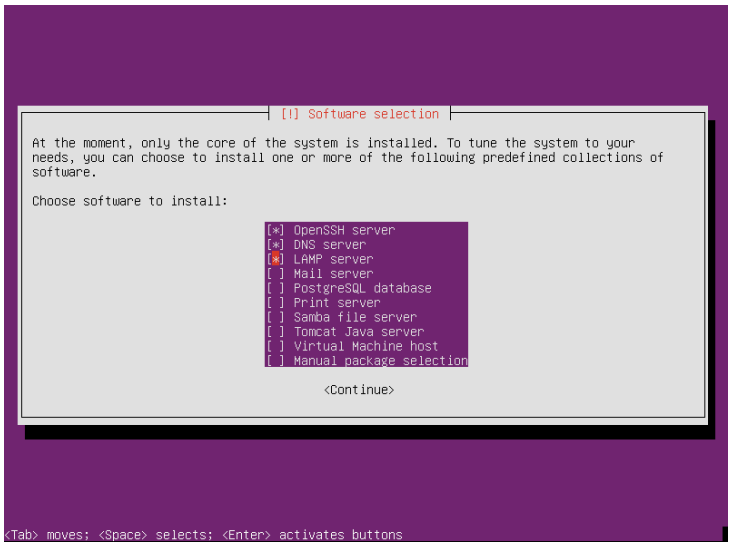
Ubuntu je besplatan OS i može se skinuti u obliku .iso datoteke putem interneta [1]. Za instalaciju .iso datoteku potrebno je snimiti na CD/DVD medij. Postupak instalacije Ubuntu Server Edition operativnog sustava jednaka je kao i kod ostalih operativnih sustava. U BIOS-u računala potrebno je CD/DVD uređaj postaviti kao "boot devices", staviti CD/DVD medij u CD/DVD uređaj i ponovo pokrenuti računalo. Prilikom pokretanja računala pokreče se program instalacije, u prvom koraku potrebno je odabrati jezik. Sljedeći se pokreče meni u kojem odabiremo "Install Ubuntu Server" (sl. 5.2).

Slika 5.2: Instalacija Ubuntu Server OS

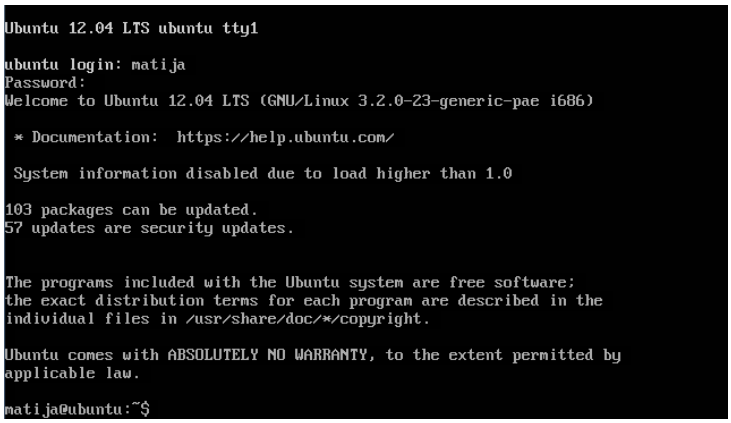
U nastavku instalacije postupak instalacije od vas traži osnovne informacije poput domene koju ćete koristiti, korisničko ime i lozinku za logiranje, "root" lozinku za MySQL. U toku instalacije moguće je odabrati određene pakete koje želimo instalirati (sl. 5.3) za naše potrebe instalirani su paketi "Open SSH server", "DNS server" te "LAMP server".

Nakon unosa osnovnih informacija pokreće se postupak instalacije koji traje nekoliko trenutaka i nakon uspješne instalacije računalo se ponovo pokreće i otvara se slijedeći prozor (sl. 5.4).

[1] http://www.ubuntu.com/download/server



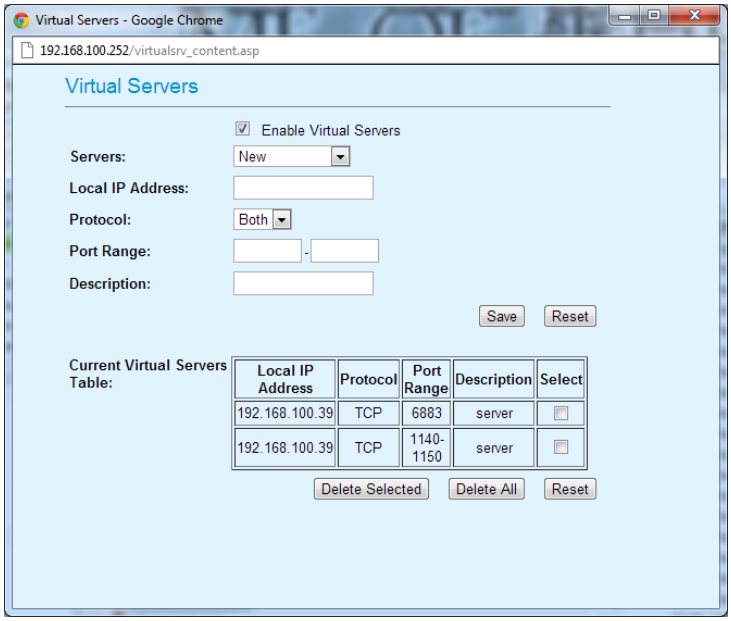
Slika 5.3: Odabir Ubuntu paketa



Slika 5.4: Uspješna instalacija

5.3 Otvaranje portova na ruter uređaju

Kako bi se moglo pristupiti server računalu sa vanjske globalne Internet mreže potrebno je na ruter uređaju otvoriti portove. U našem slučaju server računalo je spojeno na "Air Live" ruter uređaj. Ruter uređaju se pristupa preko internet pretraživača upisivanjem adrese "192.168.100.252". Odabiremo izbornik "Mode" pa "Setup" nakon toga "Virtual Server" tada se otvara forma za unos portova koje želimo otvoriti te popis portova koji su otvoreni (sl. 5.5).



Slika 5.5: Forma za otvaranje portova

Za potrebe našeg servera otvorili smo port 6883 te portove od 1140-1150. Forma se ispunjava na način da u "Local IP Adress" upišemo lokalnu adresu našeg servera gdje je vrlo bitno da server ima uvijek istu lokalnu IP adresu. U "Protocol" odabiremo "TCP" u "Port Range" upisujemo raspon portova koje želimo otvoriti i u "Description" opis npr. "server".

5.4 Podešavanje servera

5.4.1 LAMP server

LAMP [2] server je akronim kojeg čini skup besplatnih (open source) aplikacija: Linux operativni sustav, Apache web server, MySQL baza podataka i PHP programski jezik. Sve komponente LAMP servera su instalirane samo je potrebno još podesiti Apache server. Kod apache servera potrebno je promijeniti port koji će apache server slušati sa standardnog 80 na 6883 koji smo

[2] LAMP - Linux, Apache, MySQL and PHP

otvorili. To možemo učiniti tako da u datoteci `/etc/apache2/ports.conf` dvije sljedeće linije koda:

```
NameVirtualHost *:80
Listen 80
```

mijenjamo sa

```
NameVirtualHost *:6883
Listen 6883
```

Također i u datoteci `/etc/apache2/sites-available/default` mijenjamo

```
<VirtualHost *:80>
```

sa

```
<VirtualHost *:6883>
```

Nakon izvršenih promjena potrebno je ponovno pokrenuti apache2 server upisivanjem slijedeće naredbe u konzolu:

```
$ sudo /etc/init.d/apache2 restart
```

Lokacija datoteka web stranice nije mijenjana i nema je potrebe mijenjati. Datoteke web stranice stavljaju se u mapu `/var/www/`.

5.4.2 DNSdynamic

Naš je server spojen na Internet mrežu sa dinamičkom IP adresom što nam otežava pristup serveru. Svaki put kad bi htjeli pristupit serveru izvana trebali bi znati globalnu IP adresu servera što je iz praktičkih razloga nemoguće. Taj se problem može riješiti na dva načina: kupovanjem statičke IP adrese od strane internet opskrbljivača ili nekim servisom koji omogućuje dinamički dns server. Za potrebe ovog diplomskog rada znatno je jednostavnije drugo spomenuto rješenje. U rješavanju tog problema pomaže nam web servis DNSdynamic [3]. Nakon prijave na spomenuti web servis možete odabrati web adresu tipa "http://x.dnsdynamic.com" gdje je x neka riječ. Za server iz ovog zadatka odabrana je web adresa "http://arduino-matija.dnsdynamic.com". Za linux OS potrebno je instalirati "ddclient" i konfigurirati njegovu datoteku, `/etc/ddclient.conf` ,na slijedeći način:

```
daemon=60
syslog=yes
mail=root
```

[3] http://dnsdynamic.org/

```
mail-failure=root
pid=/var/run/ddclient.pid
ssl=yes

use=web, web=myip.dnsdynamic.com
server=www.dnsdynamic.com
login=username@username.com    #korisnicko ime
password=*******               #lozinka
protocol=dyndns2
arduino-matija.dnsdynamic.com
```

U korisničko ime i lozinku upisuju se podaci za prijavu u web servis "DNSdynamic". DDclient program radi na način da neprestano provjerava globalnu IP adresu web servera te kad se IP adresa promjeni on tu novu IP adresu javlja web servisu "DNSdynamic". Web servis "DNSdynamic" tada u svojoj DNS tablici mijenja IP adresu dotičnog servera.

Sada je globalna IP adresa našeg servera uvijek validna i zapisana u DNS serveru te pokazuje na web adresu "http://arduino-matija.dnsdynamic.com". Možete u pretraživač upisati adresu "http://arduino-matija.dnsdynamic.com:6883"[4] te će vam se otvorit web stranica sa web servera kojeg smo upravo podesili.

5.4.3 SSH protokol

SSH [5] je mrežni protokol koji korisnicima omogućuje uspostavu sigurnog komunikacijskog kanala između dva računala putem nesigurne računalne mreže. SSH protokol svoj rad bazira na korištenju kombinacije simetrične i asimetrične kriptografije, metoda enkripcije koje omogućuju sigurniji prijenos podataka računalnom mrežom.

SSH protokol koristit ćemo za podešavanje, kontroliranje servera sa udaljenog računala. Nakon podešavanja SSH protokola na web serveru, moći ćemo se putem programa "putty"[6] spojiti na naš web server. Te putem konzole unositi naredbe kao da smo doslovno na računalu web servera. SSH protokol se podešava u datoteci, `/etc/ssh/sshd_config`, te je potrebno zamijeniti liniju:

```
Port 22
```

Sa linijom

[4] Upisujemo 6883, jer smo podesili apache da "sluša" taj port
[5] Secure Shell
[6] http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

```
Port 1140
```

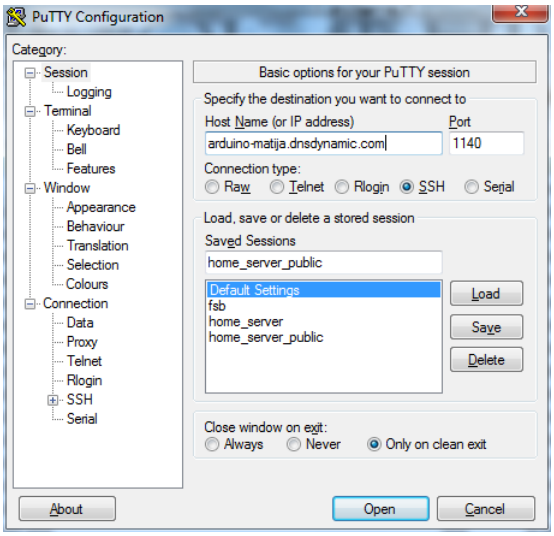
Također je potrebno dodati korisnička imena kojima želimo dopustiti spajanje na računalo na na slijedeći način:

```
AllowUsers matija ivan
```

Nakon izvršenih promjena potrebno je ponovno pokrenuti ssh protokol upisivanjem slijedeće naredbe u konzolu:

```
$ sudo /etc/init.d/ssh restart
```

Podešavanje putty programa (sl. 5.6) za spajanje na server vrlo je jednostavno. Nakon pokretanja programa upisujemo u "Host Name" u našem slučaju "arduino-matija.dnsdynamic.com", te u "Port" upisujemo 1140, pod "Connection type" odabiremo "SSH".



Slika 5.6: Podešavanje putty programa

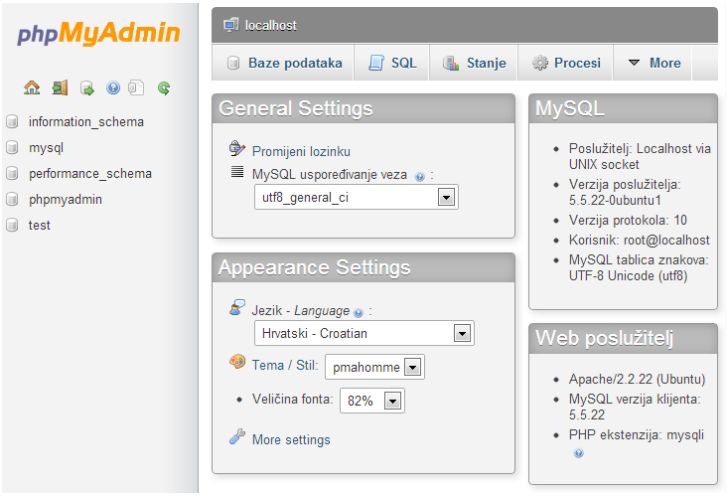
5.4.4 phpMyAdmin

Kako nam je potrebna baza podataka za spremanje podataka našeg sustava npr. temperature koristit ćemo MySQL bazu podataka. Ona je najbolji izbor jer dolazi instalirana uz Ubuntu OS. Kako bi uređivali bazu podataka koristit ćemo phpMyAdmin (sl. 5.7). PhpMyAdmin je sučelje za jednostavniji pristup i administriranje MySQL bazama putem web preglednika. Potrebno ga je instalirati na serveru upisivanjem slijedeće naredbe u konzolu:

```
sudo apt-get install phpmyadmin
```

Tokom instalacije potrebno je unijeti "root" lozinku MySQL baze koju smo definirali prilikom instaliranja Ubuntu OS. Nakon uspješne instalacije sučelje se otvara upisivanjem

`http://<hostname>/phpmyadmin` adrese u web preglednik.



Slika 5.7: phpMyAdmin sučelje

Zaključak

U ovom diplomskom radu osmišljen je sustav za mjerenje i upravljanje fizikalnim procesima na daljinu. Sustav obuhvaća jedinicu za mjerenje, jedinicu za pohranu podataka te jedinicu koja omogućava pregled podataka. Osmišljeni sustav pruža dobru osnovu za takve i slične sustave te je vrlo jednostavan za proširenje i nadogradnju. Mjerenje temperature i regulaciju temperature vrlo je lako zamijeniti sa drugim vrstama digitalnih/analognih senzora te aktuatora što sustav čini vrlo prilagodljivim. Također je moguće mjerenje/djelovanje na više kanala.

Dobra strana osmišljenog sustava je što je mjerni dio sustava potpuno odvojen od djela za obradu i prikaz podataka. Takva struktura sustava omogućuje odvojeno razvijanje jednog dijela sustava bez poznavanja drugog. Jedna od mana sustava je da sustav gubi mjerne podatke kad veza sa internetom nije uspostavljena što je potrebno ispraviti upotrebom neke data baze na strani mjernog uređaja npr. SQLite, koja bi omogućavala spremanje podataka bez veze sa internetom (lokalno) te bi se ta baza nakon uspostave internetske veze sinkronizirala sa bazom sa servera.

Tokom izrade sustava uviđena je mogućnost proširenja web sučelja sa "reverse Ajax" tehnologijom koja bi omogućila trenutno djelovanje web servera na web sučelje. Time bi se omogućilo iscrtavanje grafova i ispis podataka u realnom vremenu bez osvježavanja web sadržaja i bez dodatnog opterećivanja procesorske moći servera. Također zgodna mogućnost za proširenje sustava je "hole punching" tehnologija. To je tehnologija koju koriste mnogi poznati web servisi poput *skype*, *teamviewer*, *msn* i sl., a omogućuje pristup nekom uređaju iza vatrozida bez otvaranja porta. Takva bi nadogradnja našem sustavu omogućila potpuni "plug and play" način rada. Kako je danas sigurnost informacija vrlo važna u vidu nadogradnje također se nalazi i SSH enkripcija podataka.

Ovim radom prikazano je kako u današnje vrijeme uz pomoć open source tehnologije moguće napraviti kvalitetan sustav za nadzor i upravljanje fizikalnim procesima na daljinu. Naš je sustav vrlo napredan i pruža puno mogućnosti za nadogradnju. Omogućava nam upravljanje više mjernih stanica sa jednog mjesta što pojeftinjuje kontrolu fizikalnih procesa te to naš sustav čini vrlo atraktivnim. Takvi su sustavi za nadzor danas općenito vrlo traženi zbog visokog nivoa automatizacije procesa u različitim područjima.

Literatura

[1] Arduino, 2012.
http://arduino.cc/en/Tutorial/HomePage.

[2] Digitalno vođenje on-line, ožujak 2012.
http://laris.fesb.hr/digitalno_vodjenje/text_5-5.htm.

[3] Daniel Shiffman. *Learning processing*. Morgan Kaufmann, USA, 2008.

[4] proxml, 2011.
http://creativecomputing.cc/p5libs/proxml/.

[5] Php, studeni 2012.
http://www.php.net/.

[6] Mysql, 2012.
http://www.mysql.com/.

[7] Server | ubuntu, 2012.
http://www.ubuntu.com/business/server/overview.

[8] Linux - ubuntu porodica operativnih sustava, 2010.
http://www.informatika.buzdo.com/s798.htm#.UJ_9huR95zo.